(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

FIXED INTERVAL SMOOTHING ALGORITHM FOR AN EXTENDED KALMAN FILTER FOR OVER-THE-HORIZON SHIP TRACKING

by

William J. Galinis

March 1989

Thesis Advisor	Harold A. Titus

Unclassified

security classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified	1b Restrictive Markings
2a Security Classification Authority	3 Distribution Availability of Report Approved for public release; distribution is unlimited.
2b Declassification Downgrading Schedule	
4 Performing Organization Report Number(s)	5 Monitoring Organization Report Number(s)
6a Name of Performing Organization Naval Postgraduate School; 6b Office Symbol *(if applicable)* 62	7a Name of Monitoring Organization Naval Postgraduate School
6c Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000
8a Name of Funding Sponsoring Organization; 8b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number
8c Address *(city, state, and ZIP code)*	10 Source of Funding Numbers: Program Element No; Project No; Task No; Work Unit Accession No

11 Title *(Include security classification)* FIXED INTERVAL SMOOTHING ALGORITHM FOR AN EXTENDED KALMAN FILTER FOR OVER-THE-HORIZON SHIP TRACKING

12 Personal Author(s) William J. Galinis

13a Type of Report	13b Time Covered	14 Date of Report *(year, month, day)*	15 Page Count
Master's Thesis	From To	March 1989	101

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes

Field	Group	Subgroup

18 Subject Terms *(continue on reverse if necessary and identify by block number)* Kalman filter, Smoothing, HFDF, Storm Tracking.

19 Abstract *(continue on reverse if necessary and identify by block number)*

The performance of an extended Kalman filter used to track a maneuvering surface target using HFDF lines-of-bearing is substantially improved by implementing a fixed interval smoothing algorithm and a maneuver detection method that uses a noise variance estimator process. This tracking routine is designed and implemented in a computer program developed for this thesis. The Hall noise model is used to accurately evaluate the performance of the tracking algorithm in a noisy environment. Several tracking scenarios are simulated and analyzed. The application of the Kalman tracker to a tropical storm tracking problem is investigated. Actual storm tracks obtained from the Joint Typhoon Warning Center in Guam, Mariana Islands are used for this research.

20 Distribution Availability of Abstract [✓ unclassified unlimited] ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual Harold A. Titus	22b Telephone *(include Area code)* (408) 646-2560	22c Office Symbol 62TS

DD FORM 1473.84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete

security classification of this page

Unclassified

Fixed Interval Smoothing Algorithm for an Extended Kalman Filter for Over-the-Horizon Ship Tracking

by

William J. Galinis
Lieutenant, United States Navy
B.S.E.E., United States Naval Academy, 1983

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1989

Author: [illegible]
William J. Galinis

Approved by: [illegible]
Harold A. Titus, Thesis Advisor

[illegible]
Stephen Jauregui, Second Reader

[illegible]
John P. Powers, Chairman,
Department of Electrical Engineering

[illegible]
Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

The performance of an extended Kalman filter used to track a maneuvering surface target using HFDF lines-of-bearing is substantially improved by implementing a fixed interval smoothing algorithm and a maneuver detection method that uses a noise variance estimator process. This tracking routine is designed and implemented in a computer program developed for this thesis. The Hall noise model is used to accurately evaluate the performance of the tracking algorithm in a noisy environment. Several tracking scenarios are simulated and analyzed. The application of the Kalman tracker to a tropical storm tracking problem is investigated. Actual storm tracks obtained from the Joint Typhoon Warning Center in Guam, Mariana Islands are used for this research.

Radio direction finding, High frequency, Data smoothing [illegible] [illegible] [illegible] [illegible] approximation, [illegible] detection, Over-the-horizon targeting, [illegible]ing surface ships, Theses. (edc)

THESIS DISCLAIMER

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to express my appreciation to Prof. Hal Titus for his instructive guidance and recommendations that kept resetting the author's learning "process" when this "process" began to diverge excessively from its intended track. I also want to thank my #1 fan, my wife Diana, whose love and support made this thesis "doable" and my son Jared who kept me laughing the whole way.

I. INTRODUCTION

In 1986, then Deputy Chief of Naval Operations for Surface Warfare (OP-03), Vice Admiral Joseph Metcalf III challenged the leadership of the U.S. Navy's Surface Warfare community to reexamine the traditional concepts of surface warfare-concepts that perhaps are preventing the Navy from taking full advantage of present and future technologies. Thus began Surface Warfare's Revolution at Sea. A vital concept to the "Revolution at Sea" is the constantly expanding oceanic battle space ("up, out, and down"). [Ref. 1]

With the advent of the modern long range cruise missile, U.S. surface forces have the capability to attack enemy surface targets at 250 nautical miles. In order to make full use of this capability however, an accurate and reliable method of over-the-horizon tracking and targeting is necessary. The current methods used to target an enemy surface force at these ranges include the use of satellites, aircraft, and intelligence sources. A major drawback to using these assets is the requirement that they transmit vital targeting information to the attacking ship. A shipboard surveillance system that provides this information locally would allow the attacking forces to operate independently without relying on other sources for targeting data, data that may or may not be available when needed for a number of reasons. In addition, the ability to operate covertly, and to track and target enemy forces without divulging any targeting information, greatly enhances the probability of success of a mission. This requires a passive acquisition system. Surveillance of the high frequency spectrum using passive radio-direction finding (RDF) sensors is one method of passive long range tracking. A shipboard high frequency-direction finding system based on an extended Kalman filter with a fixed interval smoothing algorithm can be used to accurately track and target a maneuvering surface ship.

The major thrust of this thesis deals with the problem of tracking a surface ship at long ranges using lines-of-bearing obtained from radio direction-finding sensors located on two tracking ships. It is not the purpose of this research to address the multitude of problems associated with the hardware aspects of a high frequency radio direction-finding system. The basic assumption used here is that a shipboard direction-finding system is in place that provides a line-of-bearing contaminated by an additive noise process. The extended Kalman filter and the fixed interval smoothing algorithm will be used to refine the observed lines-of-bearing and improve the accuracy of the target track. The noise added to the observed line-of-bearing is an integral part of the measurement model and should reflect the noise process that would be encountered in the HF environment as accurately as possible. For the simulations conducted, this measurement noise was modeled using first a white noise model and then with a Hall noise model.

This thesis will be an extension of a previous thesis done by Lieutenant Thomas K. Bennett. The major points of that thesis are:

- The development of an extended Kalman filter shiptracking program.
- The observations used in the shiptracking program were RDF lines-of bearing.
- The position errors achieved by this program were 10-15 nautical miles.

This thesis will attempt to improve on the previous research by implementing a fixed interval smoothing algorithm and a maneuver/divergence detection scheme that uses a noise variance estimator process. The smoothing algorithm is an off-line calculation that uses all measurements taken during a time interval $0 \leq k \leq M$ to improve the estimate. By having a more accurate assessment of what the target has done in the past, we will be better able to predict ahead and estimate a target's future course, speed, and position. The computational aspects of the smoothing algorithm will be investigated as well as the types of estimation problems where the improvement due to the smoothed calculations is significant and worth the extra computational effort. The two noise models used in the simulations will be compared as to their accuracy and their advantages or disadvantages of one or the other based on the computer simulations.

The tracking of a tropical storm is a problem similar to the ship tracking problem and is discussed in Chapter 5. The major difference between the filtering applications is that the measurement data in the storm tracker are actual position coordinates given by latitude and longitude values. This leads to a linear measurement process and therefore, the linearization required in the ship tracking problem is unnecessary in the storm tracking scenario.

II. PROBLEM STATEMENT

A. GENERAL

The tracking scenario used in this thesis involves two tracking ships moving in the general direction toward a target ship. The positions of the vessels are given in x,y coordinates. The target-tracker scenario is shown in Figure 1.

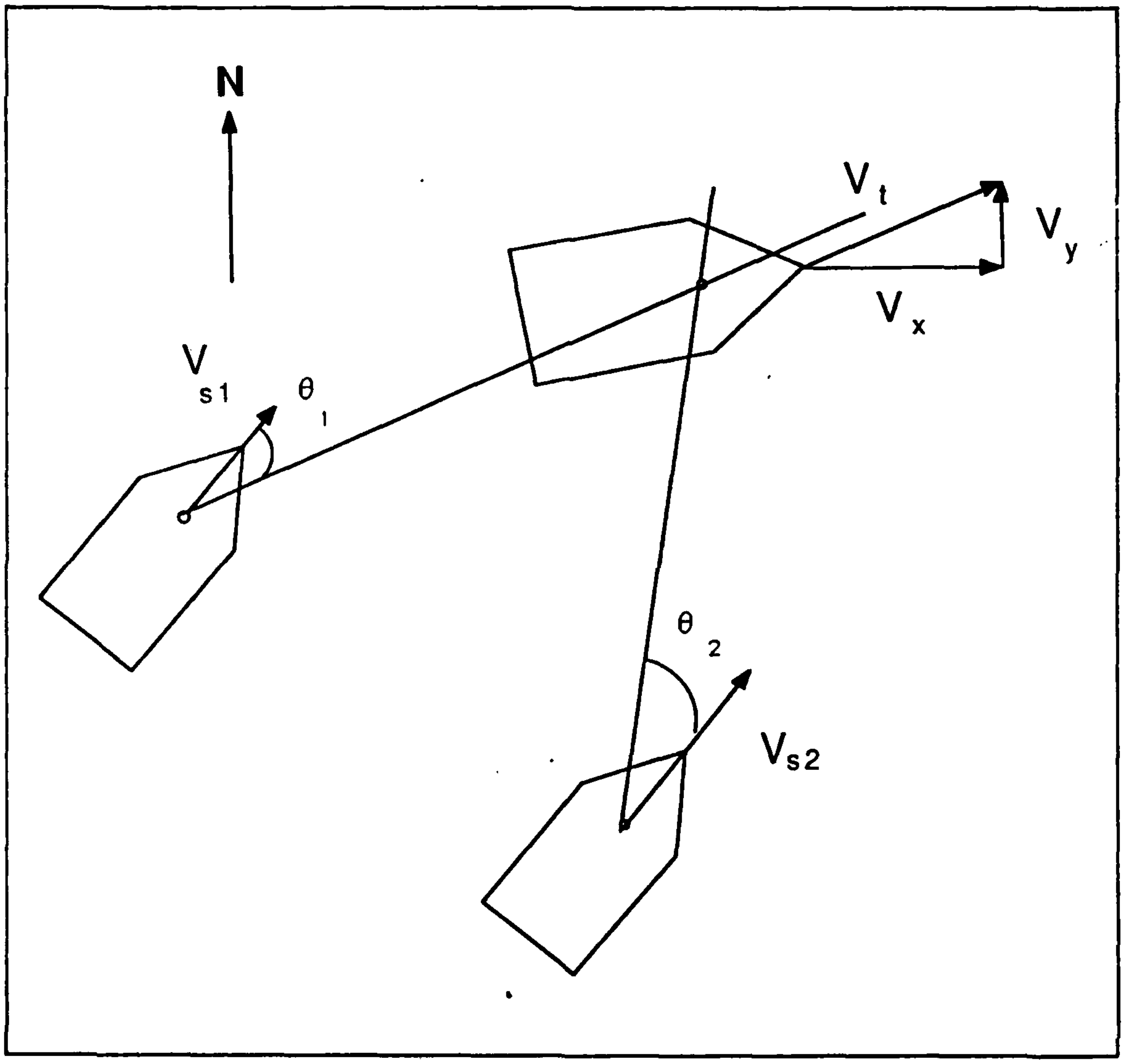

Figure 1. Surface Tracking Geometry

This problem will be developed using state space methods. Given the lines-of-bearing (the measurements) received by a radio direction-finding system, we are interested in estimating the location, course, and speed of the target (the states of the plant). The state variables for this plant are x_t, $\dot{x}_t$, y_t, and $\dot{y}_t$.

B. SYSTEM MODEL

The system to be modeled in this problem is that of a surface ship at sea. In the development of this model, the following assumptions were made.

- The effect of wind, current, and hydrodynamic forces on the ship are neglected.
- The curvature of the earth is neglected; ocean surface area is flat.
- Course and speed inputs are constant (i.e., step inputs).

This is a linear, time-distance system that can be described with the equations of motion for constant acceleration in two dimensions. The state space equation is

$$\underline{x}_{k+1} = \phi_k \underline{x}_k + \Gamma_k a_k \tag{2.1}$$

where

$\underline{x}_k =$ parameter to be estimated (state vector).

$\varphi_k =$ state transition matrix which describes how the states of the dynamic system are related

$\Gamma_k =$ system noise coefficient matrix

$a_k =$ random forcing function.

From Equation (2.1) and the above assumptions, the state vector is

$$\underline{x}_k = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} \tag{2.2}$$

and the system state equation can be expanded as

$$\begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}_k + \begin{bmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \end{bmatrix} \begin{bmatrix} a_{xk} \\ a_{yk} \end{bmatrix} \tag{2.3}$$

The system noise process for the ship tracking problem is a function of the noise coefficient matrix, Γ_k, and the random forcing function, a_k, which is simply the acceleration vector.

C. MEASUREMENT MODEL

For a linear measurement process, the measurements are linearly related to the state variables and can be modeled using the linear measurement equation

$$\underline{z}_k = H_k \underline{x}_k + \underline{v}_k \tag{2.4}$$

where

$\underline{z}_k$ = set of measurements.

H_k = observation matrix that gives the noiseless relationship between the measurements and the state vector.

$\underline{x}_k$ = state vector

$\underline{v}_k$ = measurement noise.

In this tracking problem, the measurements are the lines-of-bearing received by the radio direction-finding sensors located on two surface ships. For the geometry of the problem shown in Figure 1, the relationship of the measurements to the state variables is not linear and the measurement equation becomes

$$z_{nk} = \tan^{-1}\left[\frac{(x_{tk} - x_{nk})}{(y_{tk} - y_{nk})}\right] + v_k \tag{2.5}$$

where

z_{nk} = observed lines of bearing at time k

x_{tk}, y_{tk} = position of target at time k

x_{nk}, y_{nk} = position of sensor n at time k

v_k = measurement noise.

This nonlinear equation must be linearized prior to processing the measurement data with the filter and the smoothing algorithm. For this problem, an extended Kalman filter is required and will be discussed in the next chapter.

There are several types of noise that affect the propagation of radio signals, however, this noise can basically be divided into two categories, depending on whether it originates from within the receiving system or external to the receiving antenna. For the frequencies of interest in this problem (2-30 MHz), it is the atmospheric noise external to the receiving system that is of the greatest concern. [Ref. 2: p. 4] This noise is a function of many variables including the time of day, geographical location, season, and frequency. Although this is generally a non-white, non-gaussian noise process, it can be adequately described as a white noise process over an extended period of time. The white noise model was used for the first set of computer simulations.

A deficiency of white noise as a model for atmospheric noise, is its inability to accurately model the impulsive nature of the atmospheric noise associated with lightning discharges. This impulsive noise characteristic was investigated by Hall [Ref. 3] and led to the development of Hall's generalized "t" model which has the form

$$x(t) = m(t)n(t) \tag{2.6}$$

where $m(t)$ is a slowly varying stationary random process independent of $n(t)$ and $n(t)$ is a zero mean, narrowband Gaussian process. The complete development of this model can be found in Hall [Ref. 3].

In [Ref. 4: pp. 13-35], Spaulding outlines a method of generating random noise samples using the Hall model. The procedure for calculating these random samples starts by generating random samples from a uniform distribution, V, in the interval from zero to one and then modifying the sample according to the distribution required. For the Hall model, the random noise samples are obtained using the equation

$$X_n = \gamma\left(V^{\frac{-2}{\Theta-1}} - 1\right)^{\frac{1}{2}} \tag{2.7}$$

where $\Theta = 4$ and $\gamma = 0.707$. These values for Θ and γ were chosen using the atmospheric noise curves in [Ref. 3: p. 44]. The positive bias in the Hall model was subtracted out prior to inserting the noise data into the ship tracking algorithm. This was done in order to prevent a biased error covariance in the Kalman filter. The two noise models are shown in Figures 2 and 3. Both noise processes are zero mean and $\pm$ 3 variance.

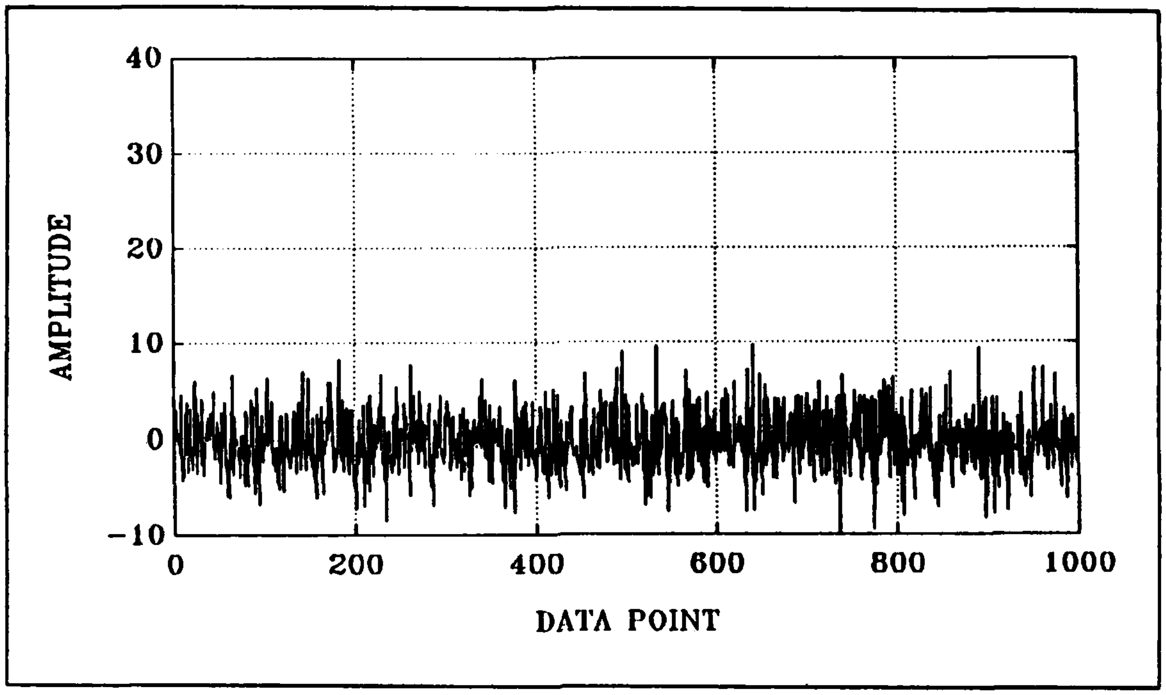

Figure 2. White Noise Model

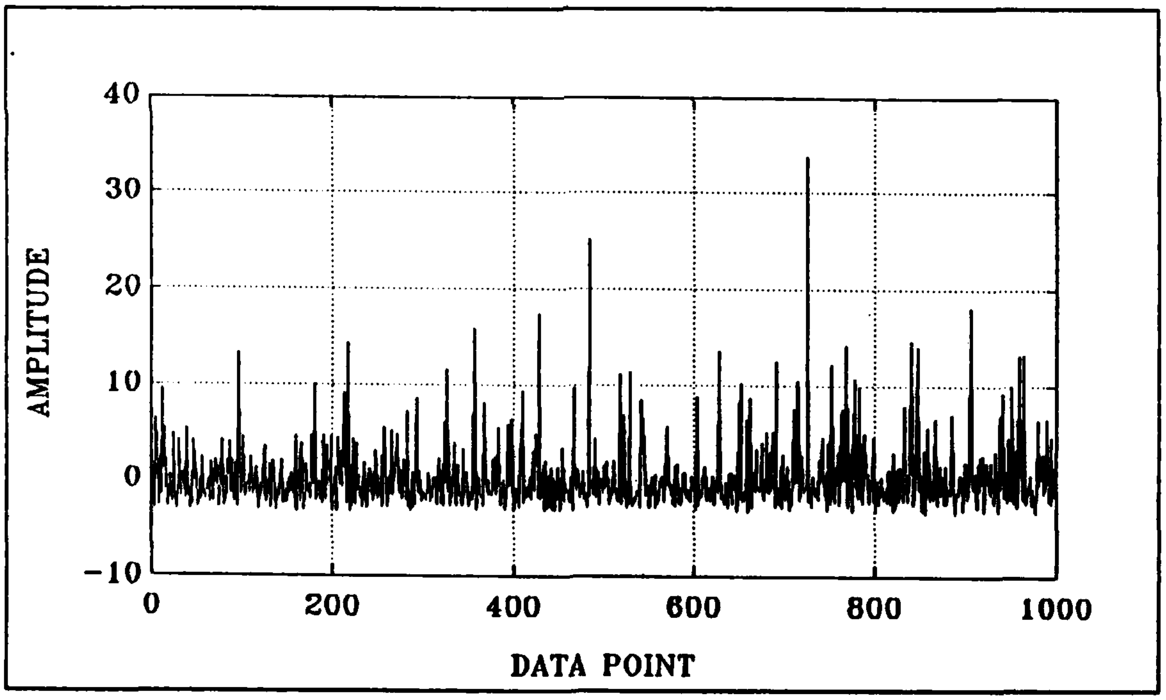

Figure 3. Hall Noise Model

III. KALMAN FILTER THEORY

A. GENERAL

Filtering refers to the process of estimating the state vector at the current time based upon all past measurements. An optimal filter concentrates on optimizing a specific performance measure used to approximate the quality of the estimate. The Kalman filter is the optimal filter in a class of linear filters that minimize the mean square estimation error between the actual and desired output. In other words, the Kalman filter attempts to minimize the elements along the main diagonal of the state error covariance matrix. The filter itself is actually a recursive algorithm for processing discrete measurements or observations in an optimal manner. [Ref. 5: p. 101] It requires a priori knowledge of the state estimate ($\underline{x}_{k-1}$) and its error covariance (P_{k-1}), and the current observation ($\underline{z}_k$). The Kalman filter is the proper algorithm to be used when both the system model and the measurement model are linear functions of the state variables and these models can be described by the equations

$$\underline{x}_{k+1} = \phi_k \underline{x}_k + \Gamma_k \underline{a}_k \qquad (3.1)$$

$$\underline{z}_k = H_k \underline{x}_k + \underline{v}_k \qquad (3.2)$$

B. EXTENDED KALMAN FILTER

From equation (2.5), we can see that there is a nonlinear relationship between the observed lines-of-bearing and the state variables. The adaptation of the Kalman filter to a nonlinear application is the extended Kalman filter. The nonlinear measurement equation is

$$\underline{z}_k = h(\underline{x}_k, k) + \underline{v}_k \qquad (3.3)$$

where the observation matrix (h_k) is a function of the state at each sampling time and the sampling index k. Linearization of this equation can be accomplished by expanding h in a Taylor series about an estimated trajectory that is continually updated with the filter's estimates. By keeping only the first term of the series expansion, a first order approximation is obtained. Higher order, more precise filters can be constructed by including more terms of the Taylor series expansion for the nonlinearities, and deriving recursive relations for the higher moments of the state vector. A detailed discussion of this procedure can be found in Gelb [Ref. 5: p. 100].

This linearization process yields the linear measurement equation

$$\underline{z}_k = H_k \underline{x}_k + \underline{v}_k \tag{3.4}$$

where

$$H_k = \left[\frac{\delta h(\underline{x}_k, k)}{\delta \underline{x}_k} \right] \tag{3.5}$$

Applying this linearization method to equation (2.5), we get

$$H_k = \frac{\delta \left[\tan^{-1} \left[\frac{(x_{tk} - x_{nk})}{(y_{tk} - y_{nk})} \right] \right]}{\delta \underline{x}_k} \tag{3.6}$$

Simplifying equation (3.6)

$$H_k = [h_{11} \quad h_{12} \quad h_{13} \quad h_{14}] \tag{3.7}$$

where

$$h_{11} = \frac{\delta \left[\tan^{-1} \left[\frac{(x_{tk} - x_{nk})}{(y_{tk} - y_{nk})} \right] \right]}{\delta x_{tk}} = \frac{(y_{tk} - y_{nk})}{\hat{R}_k^2} \tag{3.8}$$

$$h_{12} = \frac{\delta \left[\tan^{-1} \left[\frac{(x_{tk} - x_{nk})}{(y_{tk} - y_{nk})} \right] \right]}{\delta \dot{x}_{tk}} = 0 \tag{3.9}$$

$$h_{13} = \frac{\delta \left[\tan^{-1} \left[\frac{(x_{tk} - x_{nk})}{(y_{tk} - y_{nk})} \right] \right]}{\delta y_{tk}} = -\frac{(x_{tk} - x_{nk})}{\hat{R}_k^2} \tag{3.10}$$

$$h_{14} = \frac{\delta \left[\tan^{-1} \left[\frac{(x_{tk} - x_{nk})}{(y_{tk} - y_{nk})} \right] \right]}{\delta \dot{y}_{tk}} = 0 \tag{3.11}$$

By replacing y_{tk} with $\hat{y}_{t(k|k-1)}$ in equation (3.8) and x_{tk} with $\hat{x}_{t(k|k-1)}$ in equation (3.10) the linearized measurement matrix can be written as

$$H_k = \begin{bmatrix} \dfrac{(\hat{y}_{t(k|k-1)} - y_{nk})}{\hat{R}_k^2} & 0 & -\dfrac{(\hat{x}_{t(k|k-1)} - x_{nk})}{\hat{R}_k^2} & 0 \end{bmatrix} \tag{3.12}$$

where the range (R) is computed as

$$\hat{R}^2 = (\hat{y}_{t(k|k-1)} - y_{nk})^2 + (\hat{x}_{t(k|k-1)} - x_{nk})^2 \tag{3.13}$$

Having linearized the measurement process about $\hat{\underline{x}}_{(k|k-1)}$, where $\hat{\underline{x}}_{(k|k-1)}$ denotes the state estimate at time k based on all previous estimates computed at time $k - 1$, we can now use the normal linear Kalman filter equations.

C. NOISE PROCESSES

The calculation of the error covariance matrix and the filter gain matrix requires the covariance matrices for the uncorrelated noise process $\underline{a}_k$ and $\underline{v}_k$. For the measurement noise process v_k, the covariance matrix is

$$E[v_k v_i^T] = R_k \tag{3.14}$$

where R_k is defined as the state measurement noise covariance matrix. It is based on the sensor accuracy and accounts for unknown disturbances such as steps, white noise, or imperfections in the plant model. The variance of the white noise model and the Hall noise model used in the computer simulations was $\pm$ 3 degrees.

The state excitation matrix, Q_k, used in the Kalman filter represents the system noise process and is a function of the system noise coefficient matrix, Γ_k, and the random forcing function, a_k, where

$$Q_k = [\Gamma_k Q'_k \Gamma_k^T] \tag{3.15}$$

where Q'_k is defined

$$Q'_k = E[a_k a_k^T] = \begin{bmatrix} E[a_{xk}^2] & E[a_{xk} a_{yk}] \\ E[a_{yk} a_{xk}] & E[a_{yk}^2] \end{bmatrix} \tag{3.16}$$

and Γ_k is the same as in equation (2.3). The Q_k matrix allows for any random target maneuvers as well as inaccuracies in the system model. The magnitude of Q_k has a direct bearing on the magnitude of the state error covariance matrix and it prevents the covariance matrix from becoming singular by ensuring some uncertainty in the state estimates.

From Figure 1, the velocity of the target is

$$v_x = v_t \sin \Theta_t \tag{3.17}$$

$$v_y = v_t \cos \Theta_t \tag{3.18}$$

Differentiating equations (3.17) and (3.18) to get the target's acceleration

$$a_x = \dot{v}_t \left[\frac{v_x}{v_t} \right] + v_t \dot{\Theta}_t \left[\frac{v_y}{v_t} \right] \tag{3.19}$$

$$a_y = \dot{v}_t \left[\frac{v_y}{v_t} \right] - v_t \dot{\Theta}_t \left[\frac{v_x}{v_t} \right] \tag{3.20}$$

where

$$\left[\frac{v_x}{v_t} \right] = \sin \Theta_t$$

$$\left[\frac{v_y}{v_t} \right] = \cos \Theta_t$$

Here $\dot{v}_t$ and $\dot{\Theta}_t$ are equal to the acceleration along the target's course and the angular velocity or turn rate, respectively. Assuming that

$$E[\dot{\Theta}_t] = E[\dot{v}_t] = 0$$

the variances become

$$E[\dot{\Theta}_t^2] = \sigma_{\Theta t}^2 \tag{3.21}$$

$$E[\dot{v}_t^2] = \sigma_{vt}^2 \tag{3.22}$$

Approximate values [Ref. 6: p. 39] for the standard deviations of the accelerations and the angular velocity are

$$\sigma_{\Theta t} = 0.1 \ \frac{degrees}{\sec} \tag{3.23}$$

$$\sigma_{vt} = 0.01 \ \frac{knots}{\sec} \tag{3.24}$$

and the variance values are

$$\sigma^2_{\Theta t} = 0.01096\left(\frac{radians}{\min} \right)^2 \tag{3.25}$$

$$\sigma^2_{vt} = 0.0001\left(\frac{nm}{\min^2} \right)^2 \tag{3.26}$$

Taking the expectation of equations (3.19) and (3.20) and substituting in equations (3.21) and (3.22) the elements of the Q' matrix are

$$E[a^2_{xk}] = \sigma^2_{vt}\left[\frac{v_x}{v_t} \right]^2 + \sigma^2_{\Theta t}v^2_y \tag{3.27}$$

$$E[a^2_{yk}] = \sigma^2_{vt}\left[\frac{v_y}{v_t} \right]^2 + \sigma^2_{\Theta t}v^2_x \tag{3.28}$$

$$E[a_{xk}a_{yk}] = E[a_{yk}a_{xk}] = v_x v_y\left[\left(\frac{\sigma^2_{vt}}{v_t} \right)^2 - \sigma^2_{\Theta t} \right] \tag{3.29}$$

D. INITIALIZATION AND OPERATION

In the ship tracking scenario, the extended Kalman filter and the fixed interval smoothing algorithm are used to minimize the tracking errors. Prior to processing the measurement data, the filter must be initialized with an initial state estimate and an initial error covariance matrix. This initialization process is a very important step in the filter operation and gross inaccuracies in this step may cause the filter to diverge. Divergence occurs when the calculated covariance errors become much smaller than the actual covariance errors. This causes the actual values of the states to pull away from the estimated values. The concept of divergence will be discussed in greater detail in section E of this chapter. The initialization process is shown in Figure 4.

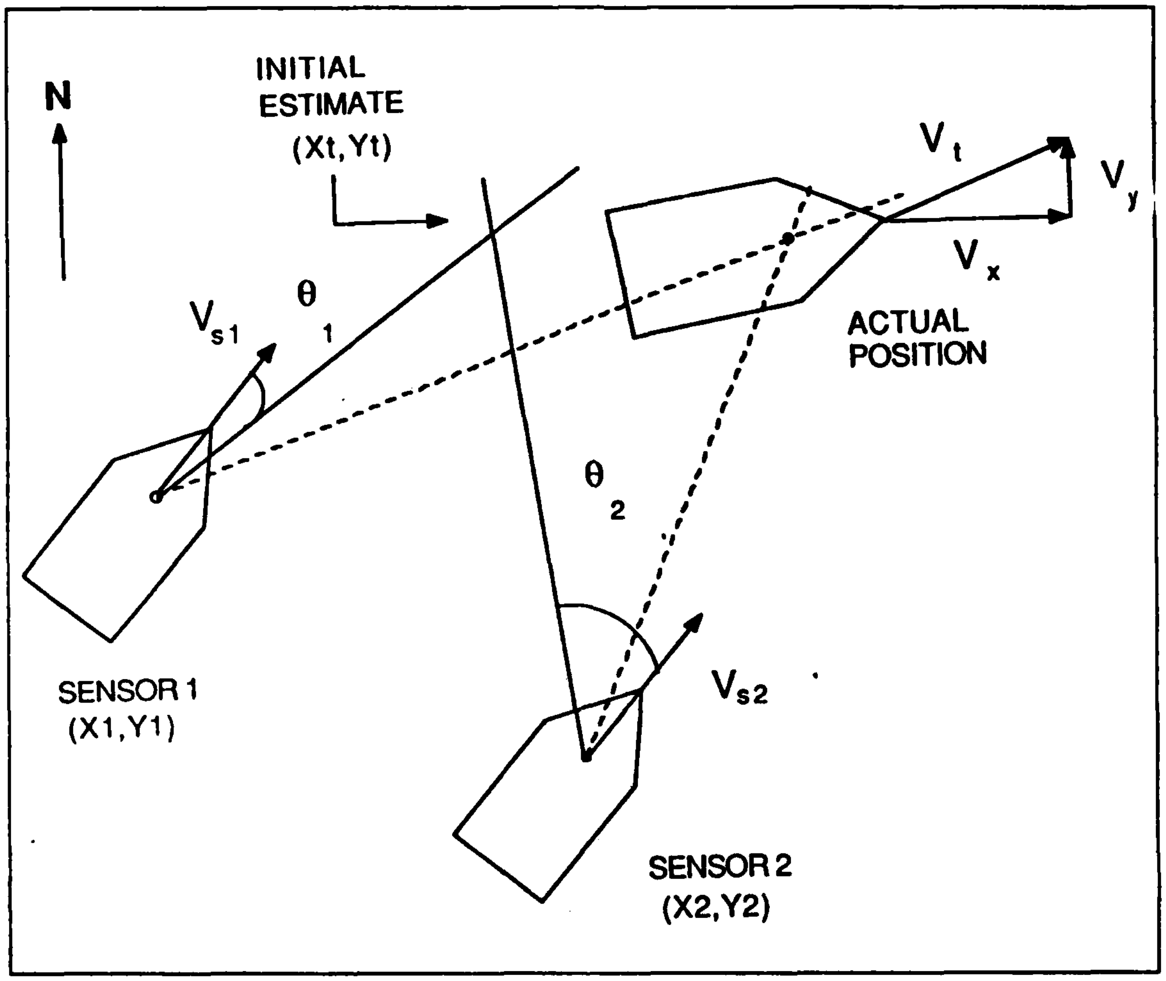

Figure 4. Initialization Process

The state estimates are the target's x and y position and the x and y components of the velocity. The initial position estimate [Ref. 7: p. 11] is the intersection of the first two lines-of-bearing received by the filter and can be calculated as

$$x_t = \left[-\frac{y_2 \tan(\theta_2) + y_1 \tan(\theta_1) + x_2 - x_1}{\tan(\theta_1) - \tan(\theta_2)} - y_1 \right] \times \tan(\theta_1) + x_1 \qquad (3.30)$$

$$y_t = -\frac{y_2 \tan(\theta_2) + y_1 \tan(\theta_1) + x_2 - x_1}{\tan(\theta_1) - \tan(\theta_2)} \qquad (3.31)$$

The initial velocity estimate is taken to be zero since there is no velocity information available to start the problem. The initial state estimates carry with them some error and it is this error or rather an estimate of this error that is used to construct the initial error covariance matrix. The initial position error is estimated to be 100 nautical miles in the x and y position and the initial velocity estimate is taken to be 0.5 nautical miles per minute or 30 knots. The errors are assumed to be zero mean and uncorrelated. Given these error approximations, the initial error covariance matrix can be written as

$$P_{(0|-1)} = \begin{bmatrix} 10000 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0 \\ 0 & 0 & 10000 & 0 \\ 0 & 0 & 0 & 0.25 \end{bmatrix} \tag{3.32}$$

The basic operation of the filter is a relatively straightforward recursive process. The equations used in the extended Kalman filter are

$$\hat{\underline{x}}_{(k|k-1)} = \phi_k \hat{\underline{x}}_{(k|k)} \tag{3.33}$$

$$P_{(k|k-1)} = \phi_k P_{(k|k)} \phi_k^T + Q_k \tag{3.34}$$

$$G_k = P_{(k|k-1)} H_k^T (H_k P_{(k|k-1)} H_k^T + R_k)^{-1} \tag{3.35}$$

$$\hat{\underline{x}}_{(k|k)} = \hat{\underline{x}}_{(k|k-1)} + G_k(\underline{z}_k - H_k \hat{\underline{x}}_{(k|k-1)}) \tag{3.36}$$

$$P_{(k|k)} = (I - G_k H_k) P_{(k|k-1)} \tag{3.37}$$

where

$\hat{\underline{x}}_{(k|k-1)}$ = projected ahead state estimate

ϕ_k = state transition matrix given by eq. (2.3)

$P_{(k|k-1)}$ = projected ahead state error covariance matrix

Q_k = state excitation covariance matrix given by eq. (3.15)

G_k = Kalman gain matrix

R_k = state measurement noise covariance matrix given by eq. (3.14)

H_k = linearized measurement matrix given by eq. (3.12).

Once the filter is initialized, we are ready to begin the data processing operation. The a priori state estimate and the state error covariance matrix are calculated using the ϕ matrix shown in equation (2.3) where T is the time difference in minutes between the observed lines-of-bearing. (It is assumed that the lines-of-bearing are received simultaneously by both sensors.) Once the projected state estimate has been calculated, it is used to calculate the linearized observation matrix in equation (3.12).

The Kalman gain matrix serves to minimize the mean square estimation error and is an indication of how much emphasis or weight will be placed on the current observation. If $P_{(k|k-1)}$ is small, the Kalman gain matrix will also be small due to the finite value of R_k. If the $P_{(k|k-1)}$ is relatively large, the gain is approximately one. By rewriting the equation for the calculation of the state estimate, equation (3.36) as

$$\hat{x}_{(k|k)} = (1 - G_k H_k)\hat{x}_{(k|k-1)} + G_k z_k \tag{3.38}$$

we can see how the Kalman gain matrix directly affects the weight placed on the current observation z_k. A large gain, indicating a large error covariance, will place more weight on the current observation as the filter tries to correct the states. A small gain, indicating a small error covariance, places less emphasis on the new observation.

If the Kalman gain is expressed as

$$G_k = P_{(k|k-1)} H_k^T R_k^{-1} \tag{3.39}$$

it can be seen that the gain matrix is "proportional" to the uncertainty in the estimate $P_{(k|k-1)}$ and "inversely proportional" to the measurement noise R_k. For a large R_k and a small $P_{(k|k-1)}$, the measurement noise (v_k) in equation (3.4) is due mainly to noise and only small corrections should be made in the state estimate. However, if R_k is small and $P_{(k|k-1)}$ is large, the measurement noise contains considerable information about the errors in the estimates and therefore a strong correction should be made to the state estimates. [Ref. 5 : p. 127-8]

E. MANEUVER AND DIVERGENCE DETECTION

The extended Kalman filter used in the tracking scenario is defined as an adaptive or self-learning filter due to its ability to manipulate the process parameters (ϕ_k, H_k, R_k, and Q_k) that change with time. If the model of the dynamic process under consideration is inaccurate, the filtered estimates tend to walk off from, or diverge from, the true estimates. As the filter locks on to the these inaccurate estimates, the state error covariance matrix gets very small which in turn causes the filter gain to decrease. When the filter gain decreases, less weight is placed on the current observations and the filter is unable to make the corrections required to correct the state estimates. The divergence problem can be detected by monitoring the filter residual process.

The residual process of the extended Kalman filter is defined as the difference between the observation at time t_k and the output of the system model based on past inputs up to time t_k. The residuals are a measure of how well the model fits the data. From equation (3.36), the residual process is

$$z_k - H_k\hat{x}_{(k|k-1)} \tag{3.40}$$

There are several techniques used to compensate for divergence. Bennett [Ref. 7: p. 14-16], observes the three most recent residual values using a moving average filter and computing the standard deviation of the residual process. It is determined that the target has maneuvered if the standard deviation exceeds a maneuver detection threshold value. The window of the moving average filter is wide enough to absorb excessive bearing errors that are far outside the standard deviation but narrow enough to detect a target maneuver soon after it occurs. This detection threshold was chosen to achieve a 90% probability of detecting a maneuver with a 10% false alarm rate.

The maneuver and divergence technique implemented in this thesis is an adaptive noise estimation technique presented by Jazwinski [Ref. 8: pp. 311-315], and utilized by Olcovich [Ref. 9: pp. 30-33]. This technique examines the residual of each observation and compares the residual value to an adaptive gate where the adaptive gate is defined as three times the predicted residual standard deviation. Defining the variance of the residual [Ref. 8: p.271] as

$$r_{(k|k+1)} = H_k P_{(k|k-1)} H_k^T + R_k \tag{3.41}$$

The predicted residual standard deviation is

$$\sigma_k = \sqrt{r_{(k|k+1)}} = \sqrt{H_k P_{(k|k-1)} H_k^T + R_k} \tag{3.42}$$

and the adaptive gate becomes

$$GATE = 3\sigma_k \tag{3.43}$$

For each observation, the residual is compared to the adaptive gate. If the residual value is less than the value of the adaptive gate, the filter continues on and processes the next observation. If the residual value is greater than the adaptive gate, the divergence detection and compensation algorithm begins.

The state excitation matrix, Q_k, is increased by increasing the elements along the main diagonal of the Q'_k matrix given by equation (3.16). The old a priori value of the error covariance matrix, $P_{(k/k-1)}$, given by equation (3.34), is increased by adding the new state excitation matrix to it. A new filter gain, G_k, and a new predicted residual variance, $r_{(k/k-1)}$, are then calculated using equations (3.35) and (3.41) respectively. By increasing Q_k and adding it to $P_{(k/k-1)}$, we prevent the state error covariance from becoming overly optimistic while at the same time increasing the Kalman filter gain to appropriately weigh the current observations. The larger predicted residual variance increases the width of the adaptive gate and opens the filter window to a larger bearing deviation. If this bearing deviation exceeds the adaptive window for three consecutive iterations using the same observation, the filter determines that the target has maneuvered and the filter parameters are reset.

In order to efficiently implement this maneuver/divergence detection process, the value of Q_k that is added to $P_{(k/k-1)}$ must be carefully evaluated. If Q_k is increased too much, $P_{(k/k-1)}$ will grow without bound resulting in a highly unstable filter. This value of Q_k should be sufficiently high in order to open the adaptive gate wide enough to account for a random noisy bearing without indicating a target maneuver, and yet low enough so that a maneuvering bearing will be outside the filter window and the maneuver will be detected. A more sensitive filter, one with a narrow adaptive gate that would be exceeded rather frequently by noisy bearings, will give a higher number of "false alarms" or indications that the target has maneuvered when it actually has not. On the other hand, a less sensitive filter will tend to "miss" a target maneuver more often, due to the larger window of the adaptive gate. However, it will also provide fewer erroneous target maneuver indications. The idea is to design the adaptive gate parameters in order to optimize the number of actual target maneuver detections, to minimize the number of erroneous indications, and to prevent the filter from becoming unstable.

For the ship tracking scenario, the state excitation matrix Q_k was increased by increasing the coefficients along the main diagonal of the Q'_k matrix by a factor of 2.5. These coefficients account for the random course and speed changes of the target. The multiplicative constant of 2.5, found by trial and error, increases the width of the adaptive gate by three percent for each iteration.

The advantage of using this divergence algorithm over the moving average process used in Ref. 7, is that a target maneuver can be detected with one observation as opposed to three observations required in the previous filtering algorithm. This advantage also poses a serious defect in that the divergence/maneuver detection decision is based on only one residual and, therefore, has little statistical significance. This can be corrected with the smoothing algorithm. [Ref. 8 : p. 313]

F. SMOOTHING ALGORITHM

Smoothing is an off-line procedure that uses all the state estimates produced by an estimator and attempts to improves the accuracy of these estimates by using more measurements to produce the smoothed estimate. The estimator used in this thesis is the extended Kalman filter described above. The basic idea behind smoothing is that for a time interval from 0 to K, an estimate at time *k* based on all previous estimates up to time *K*, $(\hat{x}_{(k|K)})$, will be more accurate than an estimate based only on the estimates up to time *k*, $(\hat{x}_{(k|k)})$.

Meditch [Ref. 10: p. 193] categorizes smoothing algorithms into three particular groups.

Fixed Point Smoothing smooths the estimate $\hat{x}_{(k|K)}$ at a fixed point *k* as *K* increases.

Fixed Lag Smoothing smooths the estimate $\hat{x}_{(K-N|K)}$ for a fixed delay *N* as *K* increases.

Fixed Interval Smoothing smooths the estimate $\hat{x}_{(k|K)}$ over the time interval from 0 to K where K is fixed and *k* varies from 0 to K.

This thesis uses a fixed-interval smoothing algorithm [Ref. 10: p. 216-224], to smooth the state estimates of the extended Kalman filter in the ship and storm tracking scenarios. This smoothing routine provides the optimal state estimate at each time *k* over a fixed interval from 0 to K. The smoothing algorithm is entered with the a priori and a posteriori estimates and their associated covariance matrices. The equations used in the smoothing algorithm are

$$A_k = P_{(k|k)}\Phi^T P^{-1}_{(k+1|k)} \tag{3.44}$$

$$\hat{x}_{(k|N)} = \hat{x}_{(k|k)} + A_k(\hat{x}_{(k+1|N)} - \hat{x}(k+1 \mid k)) \tag{3.45}$$

$$P_{(k|N)} = P_{(k|k)} + A_k(P_{(k+1|N)} - P_{(k+1|k)})A_k^T \tag{3.46}$$

where

A_k = smoothing filter gain matrix

$\hat{x}_{(k|N)}$ = smoothed state estimate a time k based on N observations

$P_{(k|N)}$ = smoothed state error covariance matrix.

At the start of the smoothing routine, the last filtered estimate becomes the first smoothed estimate. The index k in equations (3.44-46) is decremented by one for each pass through the smoother with the beginning value of k equal to the number of data points to be smoothed, minus one ($N - 1$). Consequently, the program makes $N - 1$ passes through the smoothing algorithm.

IV. COMPUTER SIMULATIONS

A. GENERAL

The SHIPTRACK extended Kalman filter program used in Ref. 7 was originally implemented on an Apple Macintosh Plus microcomputer. This program was modified and adapted to run on an IBM PC. A complete listing of the SHIPSM.FOR extended Kalman filter and fixed interval smoothing algorithm is included in Appendix A. The general scenario used in all of the computer simulation cases is that of two tracking ships moving in the general direction of the target ship. The track data required by SHIPSM.FOR for each scenario was generated using the program [Ref. 7] TRACKDATA.FOR. This program calls for the initial course, speed, and position of the two tracking ships and the target ship, the time of each HFDF interception, and any course and speed changes of the target. (The course and speed of the tracking ships is held constant throughout each simulation run.) The atmospheric noise is added to the observed HFDF line-of-bearing in the TRACKDATA.FOR routine. It generates an output file called TRKDATA.DAT that contains the time of each observation, the position of each tracking ship at each observation, and the angle of reception of each HF intercept. This is the input file for the filter and smoothing algorithm. SHIPSM.FOR generates four output files. FILDATA.DAT and SMDATA.DAT contain the track information and the files ELLIP.DAT and ELLIPS.DAT contain the data required to plot the error ellipses for the filtered and the smoothed positions respectively. The error ellipses provide a graphical representation of the accuracy of the estimate.

The error ellipses generated [Ref. 7: pp. 16-17] by the filter and smoothing algorithm represent an area where the probability of the target's true position being within the ellipse is 68%. The error ellipses are plotted for every fourth position on the overall track plot for each scenario (if the ellipses are plotted for every point the plots become too cluttered to be interpreted accurately). A smoothed observation generates a smaller error ellipse than the filtered estimate. This is due to the smaller error variances for the smoothed estimate compared to the error variances calculated for the filtered estimates and reflect the higher degree of confidence placed in the accuracy of the smoothed estimate. The last filtered estimate is equal to the first smoothed estimate and, therefore, the smoothed and filtered error ellipse for this point overlap.

Graphical results were obtained using the Matlab graphics package and the plots included are representative of the results obtained from the different tracking scenarios. The first graph is a geographical plot that compares the observed track, or the raw data, with the filtered and smoothed track data and is used to demonstrate the effect of the Kalman filter and the smoothing algorithm on the noisy data. The following two graphs present the position errors and the variance in the *x* direction associated with the track data. The last plot is included to give the reader an idea of how accurate the filtered and the smoothed tracks are when compared to the target's true track.

The computer simulations consisted of nine targeting scenarios as listed below.

- Scenario #1--maneuver toward tracking ships, no measurement noise
- Scenario #2--no maneuver, white noise model
- Scenario #3--no maneuver, Hall noise model
- Scenario #4--maneuver toward tracking ships, white noise model
- Scenario #5--maneuver toward tracking ships, Hall noise model
- Scenario #6--maneuver away from tracking ships, white noise model
- Scenario #7--maneuver away from tracking ships, Hall noise model
- Scenario #8--2 maneuvers away from tracking ships, white noise model
- Scenario #9--2 maneuvers away from tracking ships, Hall noise model

Scenarios one thru seven were started with the target ship at a position of (-75,150), tracking ship #1 at a position of (-30,0), and tracking ship #2 at a position of (30,0). For scenarios eight and nine, the target ship started at the origin and the tracking ships were located at the above positions. The tracking ships were set on a course of 000° at 10 knots for each scenario. The speed of the target ship was held constant at 15 knots throughout the simulations. The data was collected over a ten hour time frame with the HF intercepts recorded at 30 minute intervals for a total of 21 intercepts for each simulation. All times are given in minutes.

The percent improvement in the position error due to the Kalman filter and the smoothing algorithm, is given for the different scenarios. This percentage was calculated by taking the percent of improvement, ($\pm$), for each observation and then averaging these percentages over the time interval of interest. These values give a relative indication of the effectiveness or ineffectiveness of the filter and the smoother for a particular scenario.

B. SCENARIO #1

The first tracking scenario was used to verify that the Kalman filter and the smoothing algorithm would accurately track a maneuvering target in a noiseless environment. The results for this scenario are shown in Figures 5-8. In this case, the observed track equals the true track due to the absence of noise in the bearing measurements. The initial track error shown in Figure 6 is due to the error in the initial state estimates. When the target maneuvers at 300 minutes, the tracking error increases dramatically for the first observation after the turn, however, it returns to zero two observations later as the filter regains the target track.

The error ellipses in Figure 8 demonstrate graphically how the accuracy of the position estimates increases as the problem progresses. At time zero, the tracking ships are approximately 150 nm southeast of the target. The large distance in the y direction is reflected in the size and orientation of the major axis of the ellipse lying in a southeasterly direction. As the tracking ships move north, the magnitude of the major axis decreases and the direction of this axis rotates as the target ship passes in front of the advancing tracking ships and eventually to the east of at a range of approximately 10 nm. At the end of the scenario, the tracking ships are located northeast of the target.

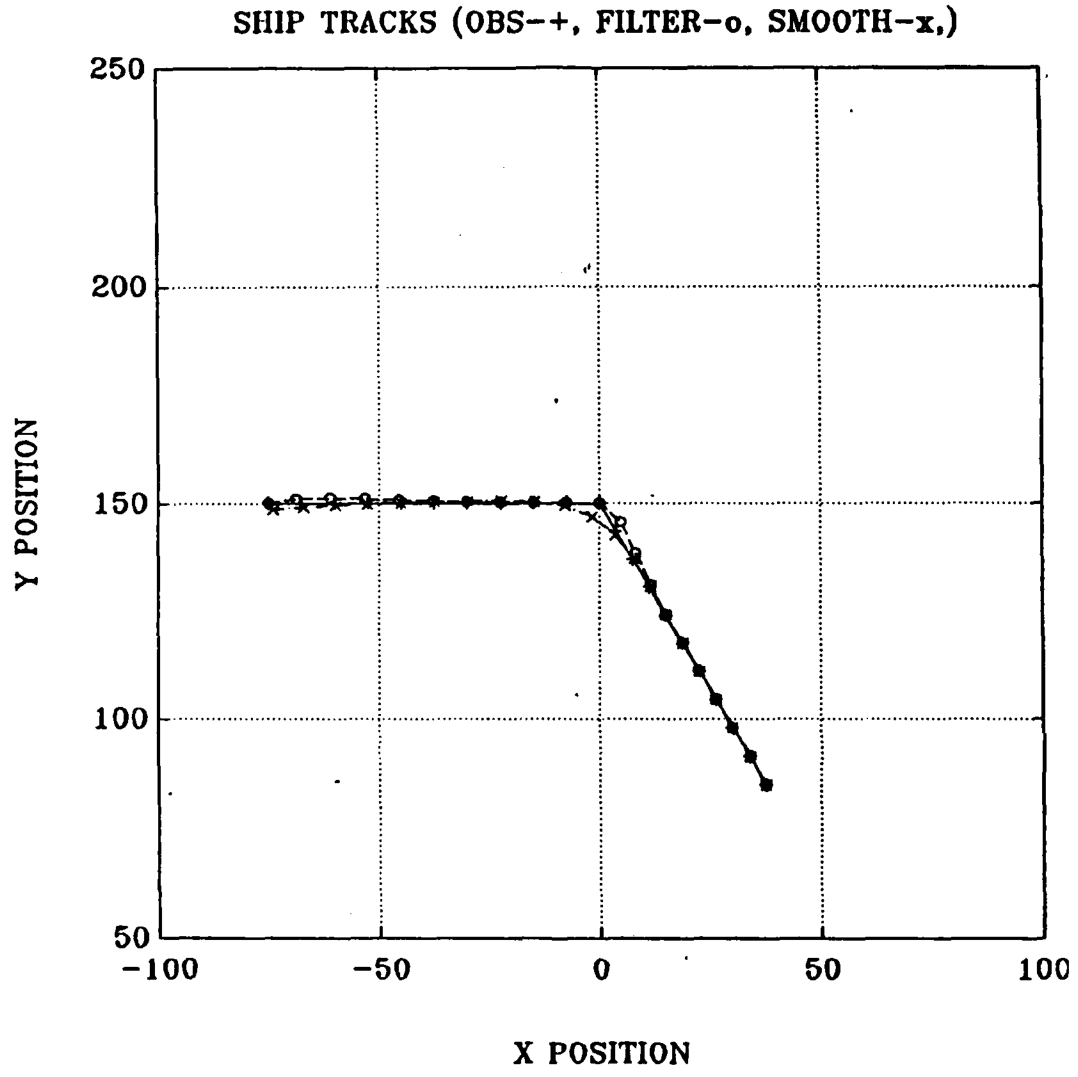

Figure 5. Processed Data vs Observed Results

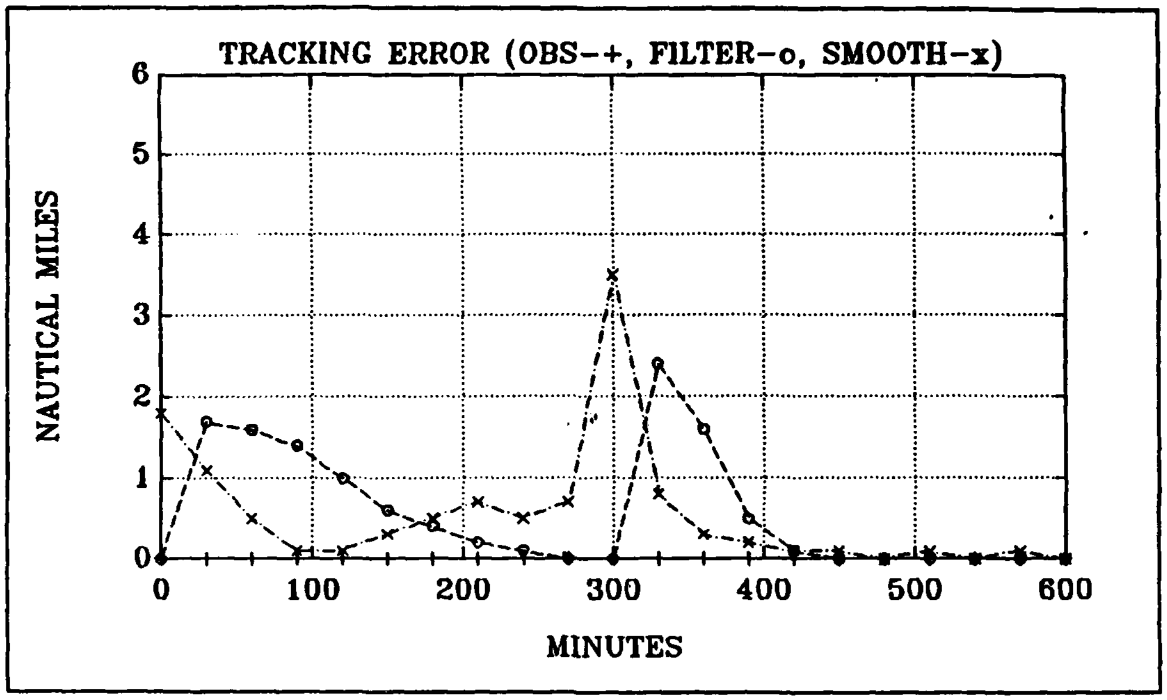

Figure 6. Position Error

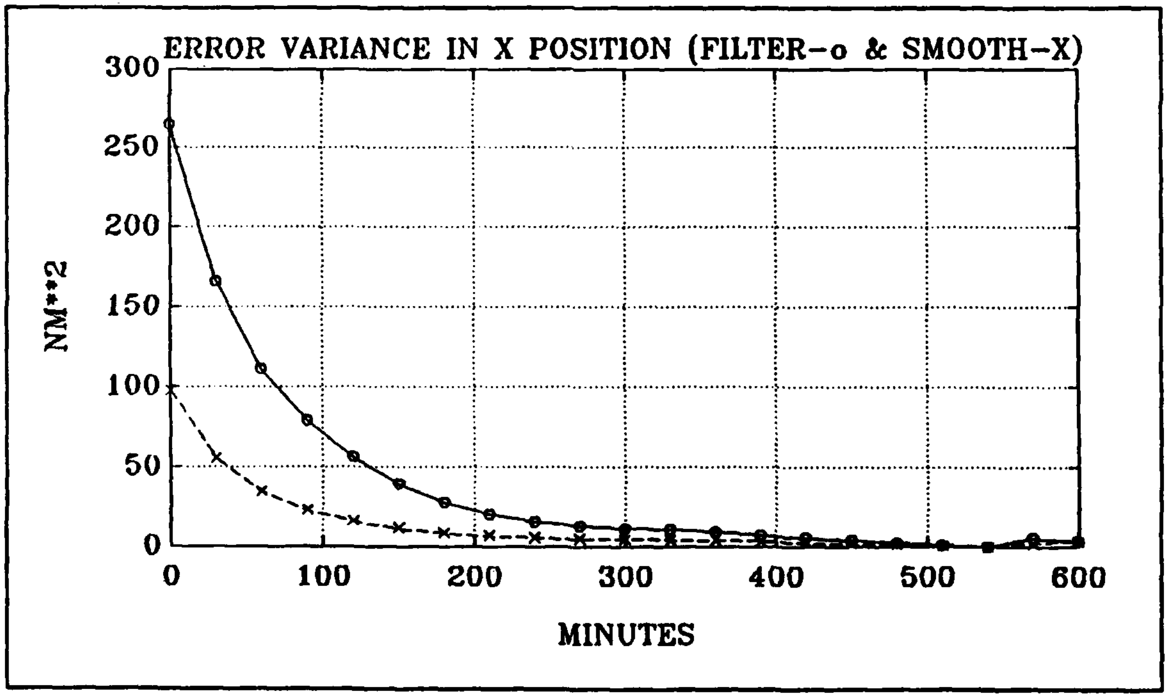

Figure 7. Variance of Position Error in the X Direction

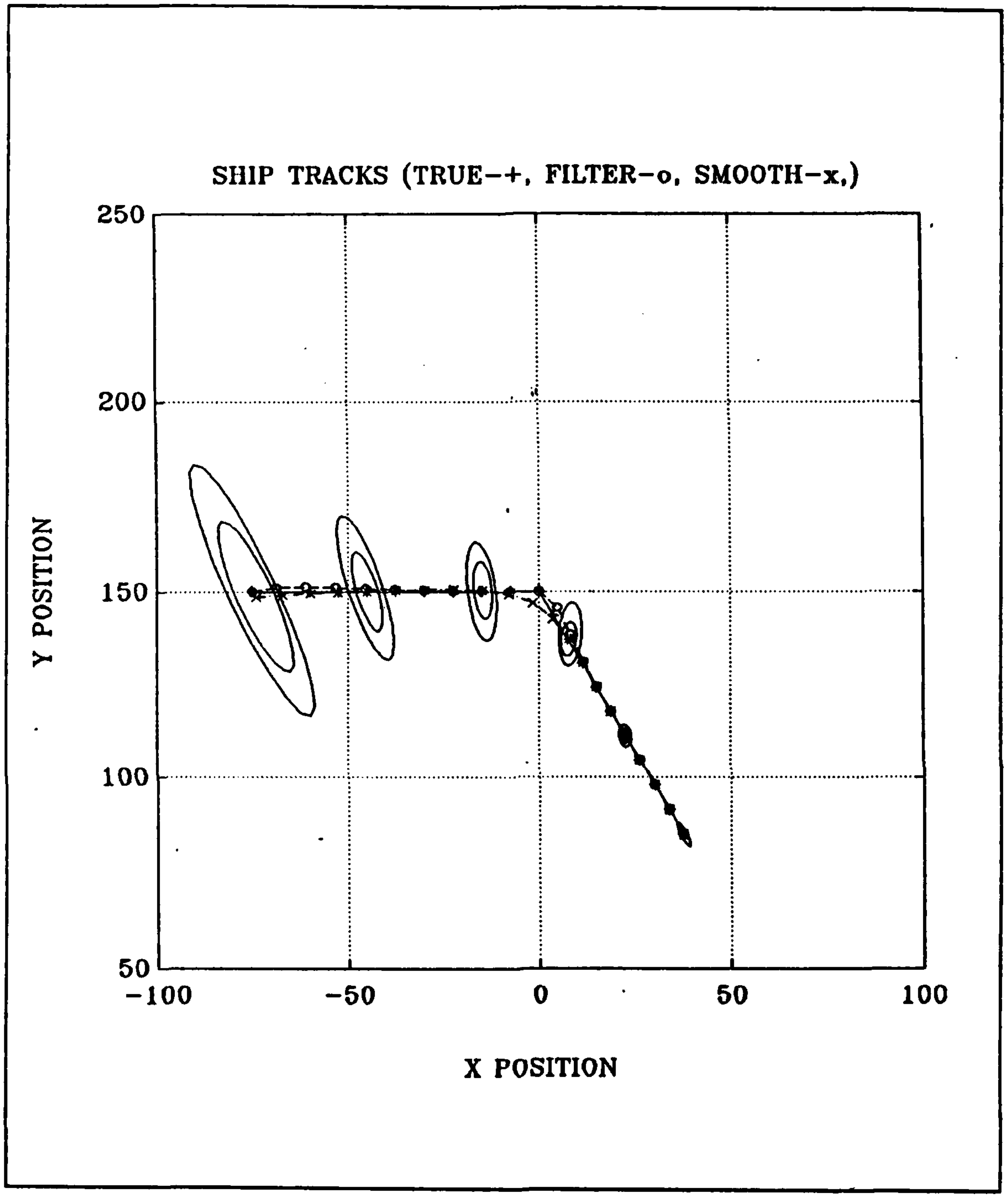

Figure 8. Scenario #1 Overall Track Results

C. SCENARIO #2

In this case, the target is steaming due east at 15 knots for the entire simulation. The results for this scenario are shown in Figures 9-12. The HFDF lines-of-bearing are distorted using a white noise model to represent the atmospheric noise. The observed track, calculated by taking the intercept of the unfilterd lines-of-bearing, is shown in Figure 9. From this plot and the error plot of Figure 10, we can see how the extended Kalman filter and the fixed interval smoothing improve the overall track estimate and decrease the position error. The Kalman filter improves the position accuracy by an average of 22% and the smoother improves the position accuracy by an average of 53%. The filtered target speed is 19 knots and the smoothed target speed is 14.8 knots. The increased confidence in the accuracy of the smoothed estimates is reflected in the plot of the error variance in the x direction. Figure 12 shows the relationship of filtered and smoothed tracks to the true target track.

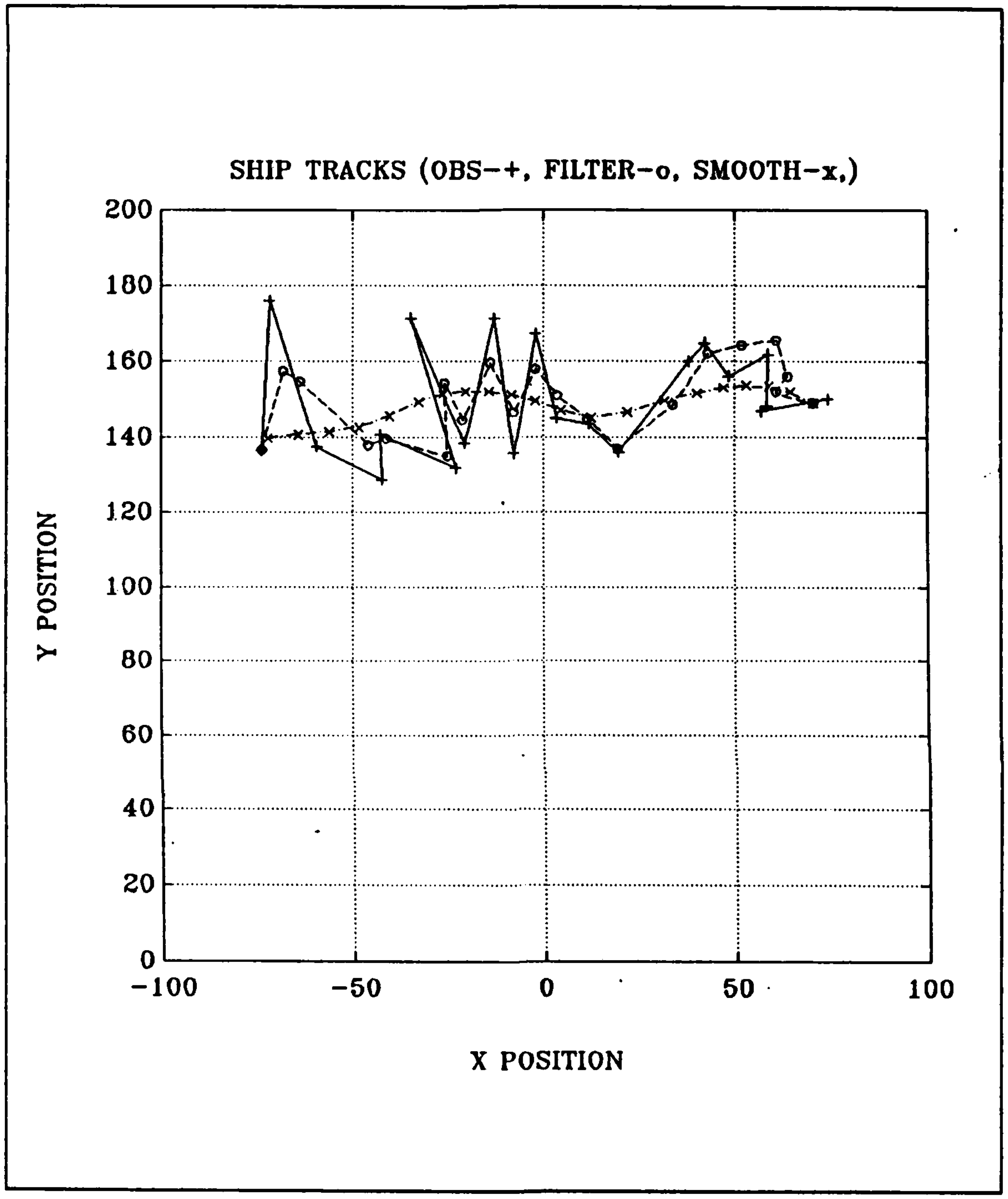

Figure 9. Processed Data vs Observed Results

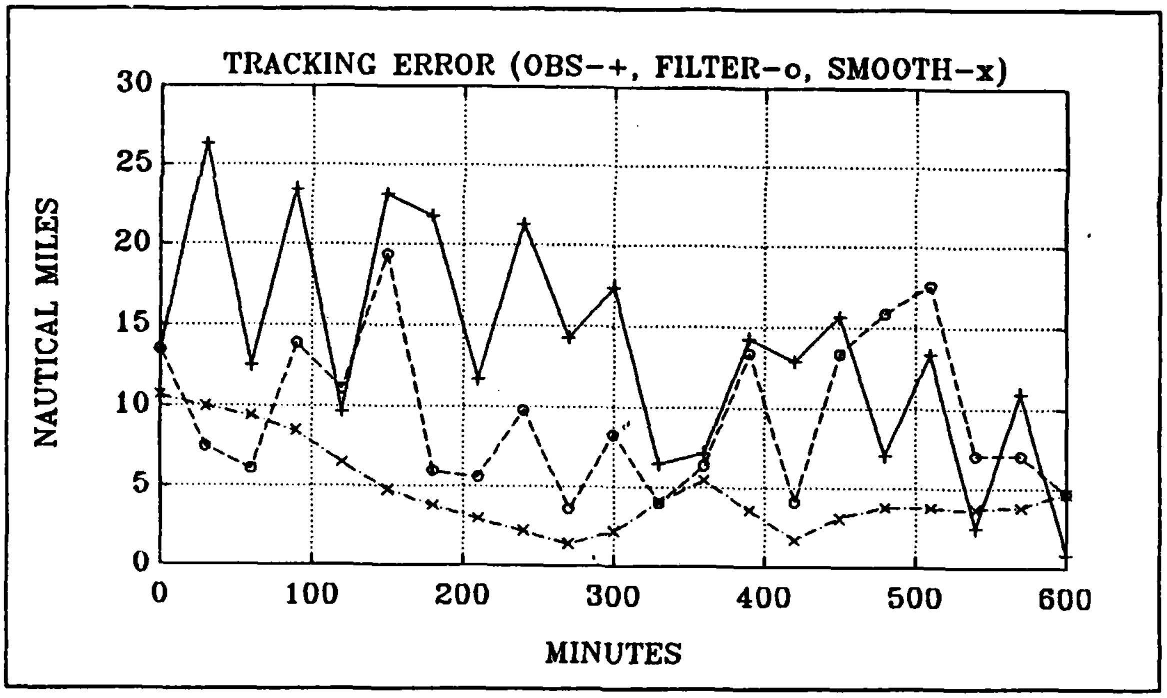

Figure 10. Position Error

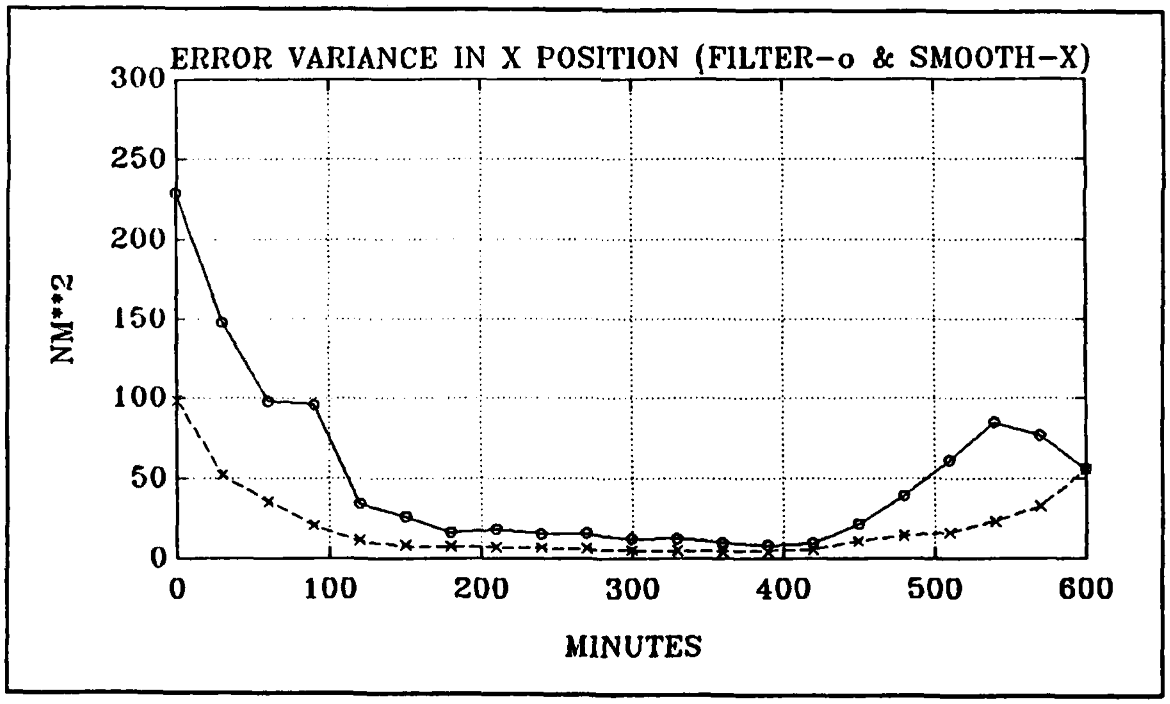

Figure 11. Variance of Position Error in the X Direction

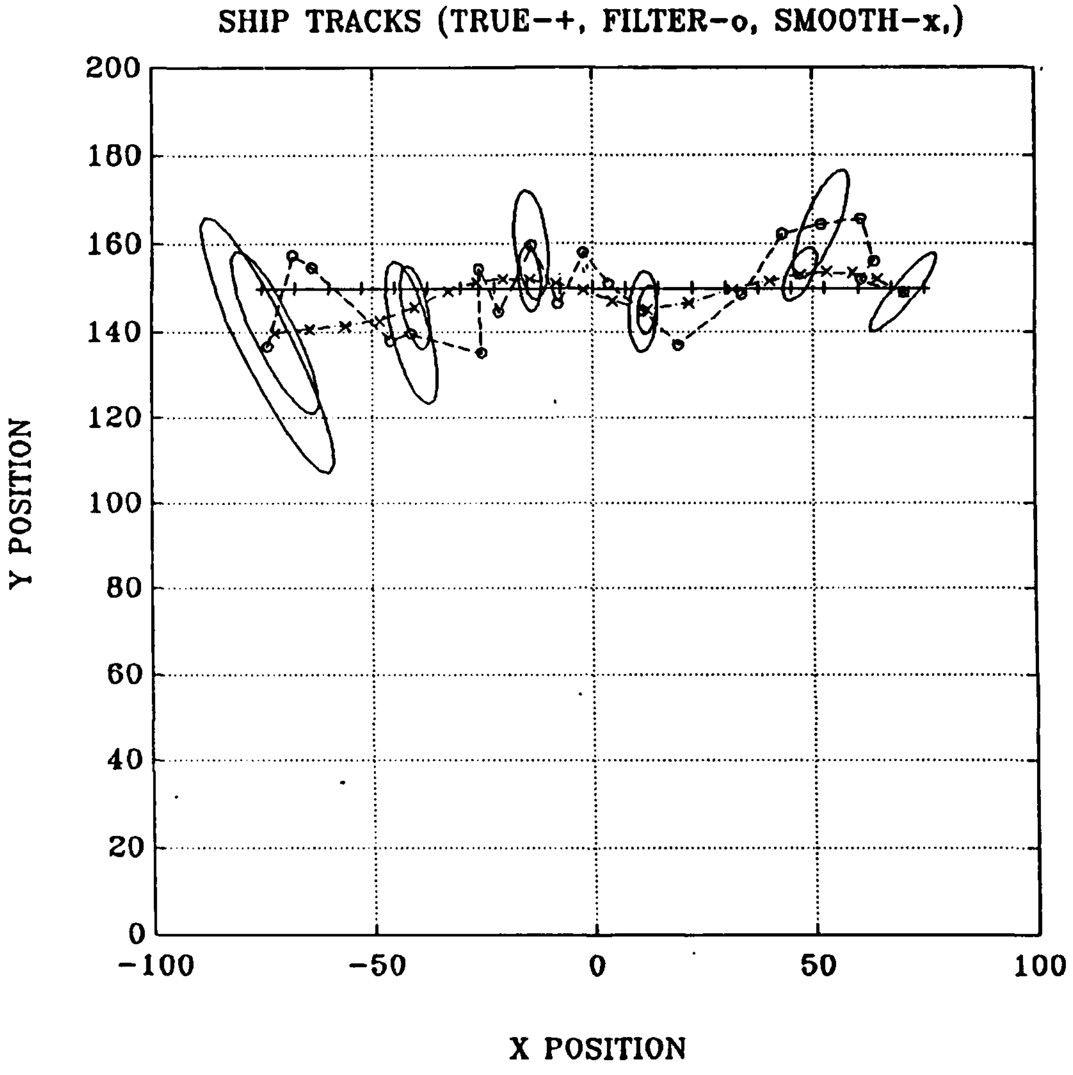

Figure 12. Scenario #2 Overall Track Results

D. SCENARIO #3

This scenario is the same as the previous one except that the white noise model is replaced by the Hall noise model. The results for this scenario are shown in Figures 13-16. The effects of the impulsive characteristics of this noise model are evident in the early part of this simulation. A "wild" bearing, due to a noise spike, occurs at time 90 resulting in a large position error. This causes the next three filtered estimates to be very inaccurate, however, the filter does begin to correct its track two observations after the noise spike occurred. It is interesting to see that the smoother just about eliminates the large tracking errors due to the erratic behavior of the Kalman filter. During the time interval from 90 to 240 minutes, the average position error of the filtered estimates is more than 850% worse than the observed position error while the average smoothed position error is less than 130% of the observed error. Even with this improvement, the position error is still greater than 20 nm and unacceptable for a long range missile attack. As the simulation progresses and the ships move toward each other, the track solution is refined and the track error remains inside of ten miles. The small noise spike at 570 minutes is completely smoothed by the smoothing algorithm. The average target speed estimated by the filter is 20.3 knots and the smoother refines that estimate to 17 knots.

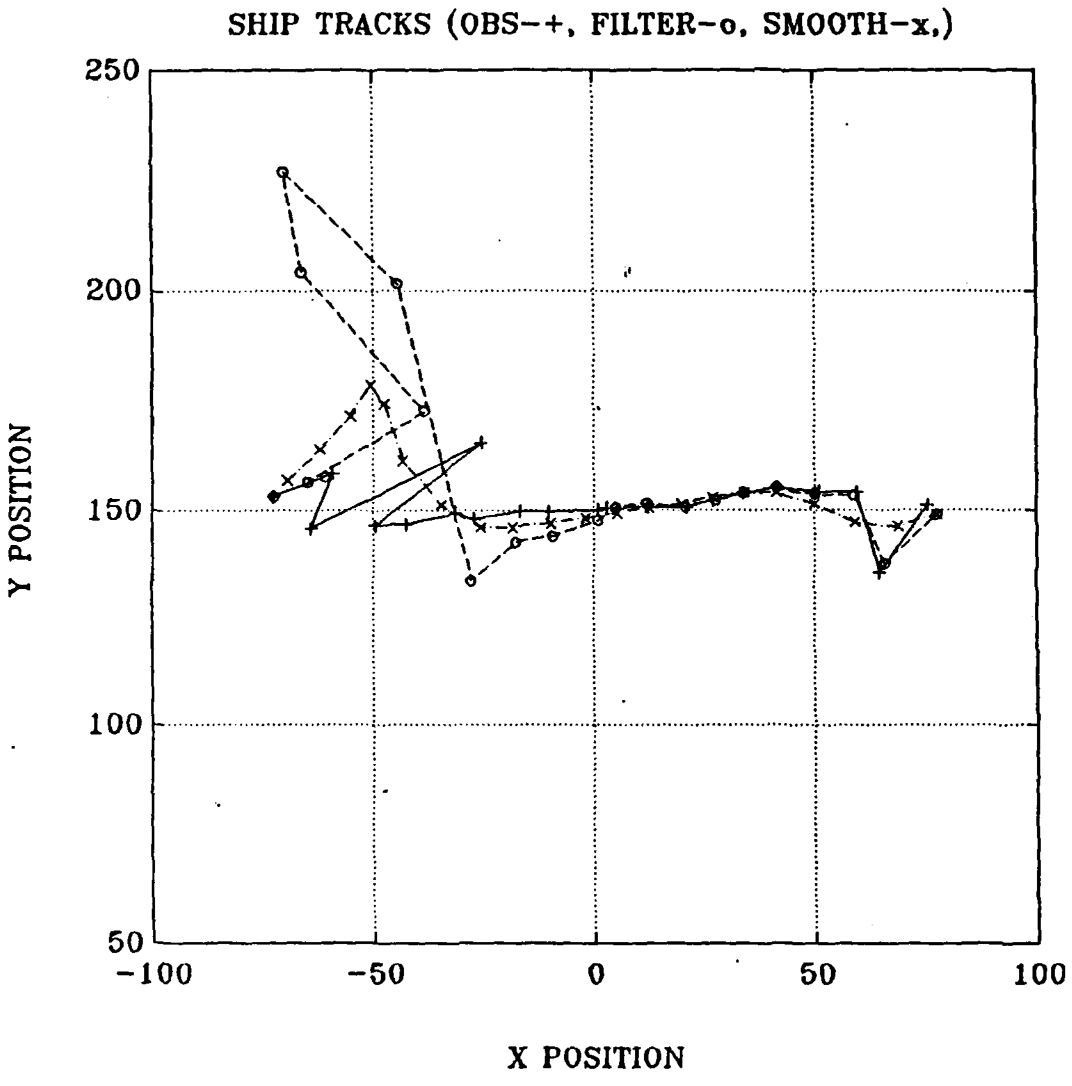

Figure 13. Processed Data vs Observed Results

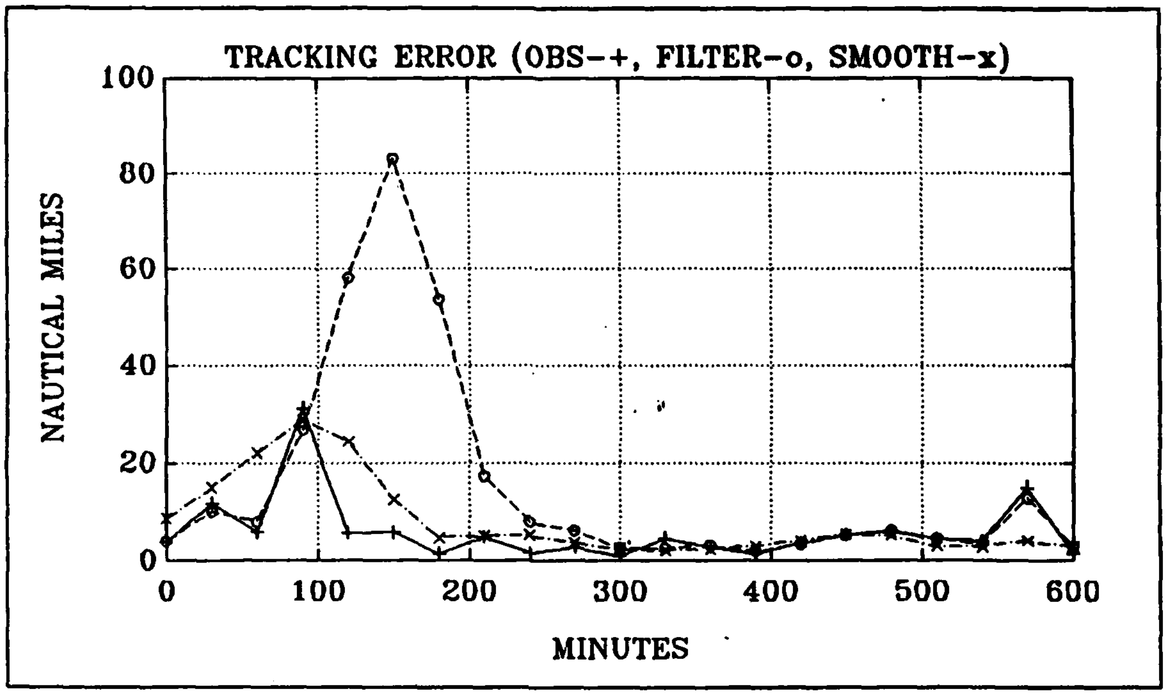

Figure 14. Position Error

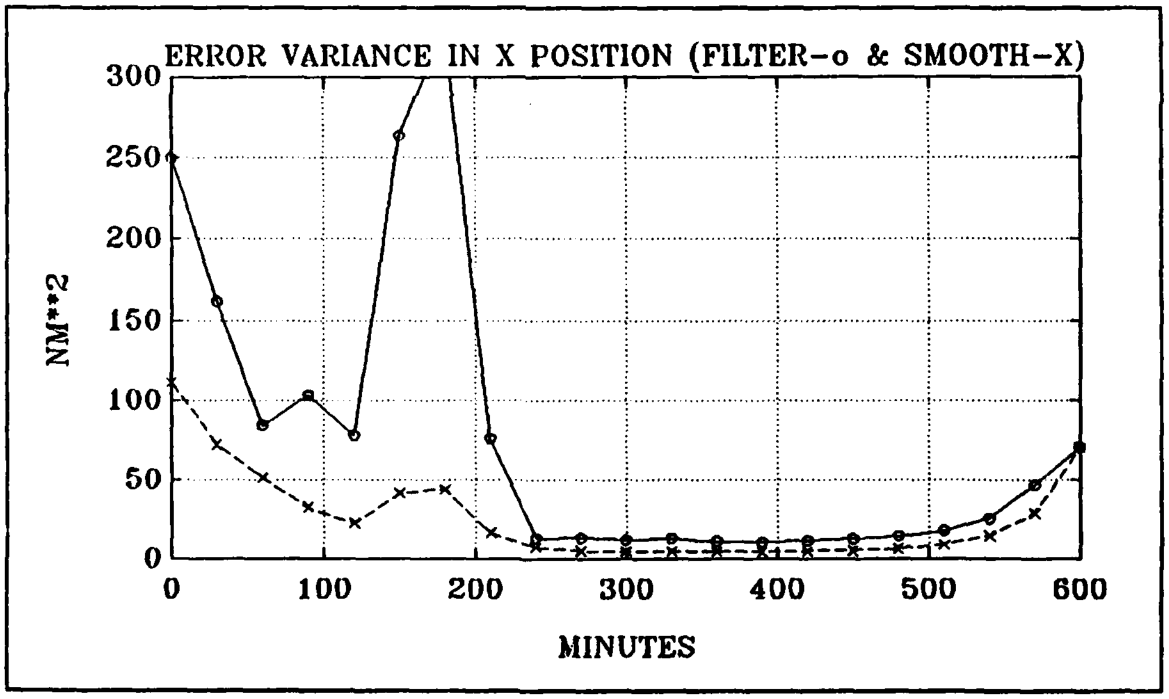

Figure 15. Variance of Position Error in the X Direction

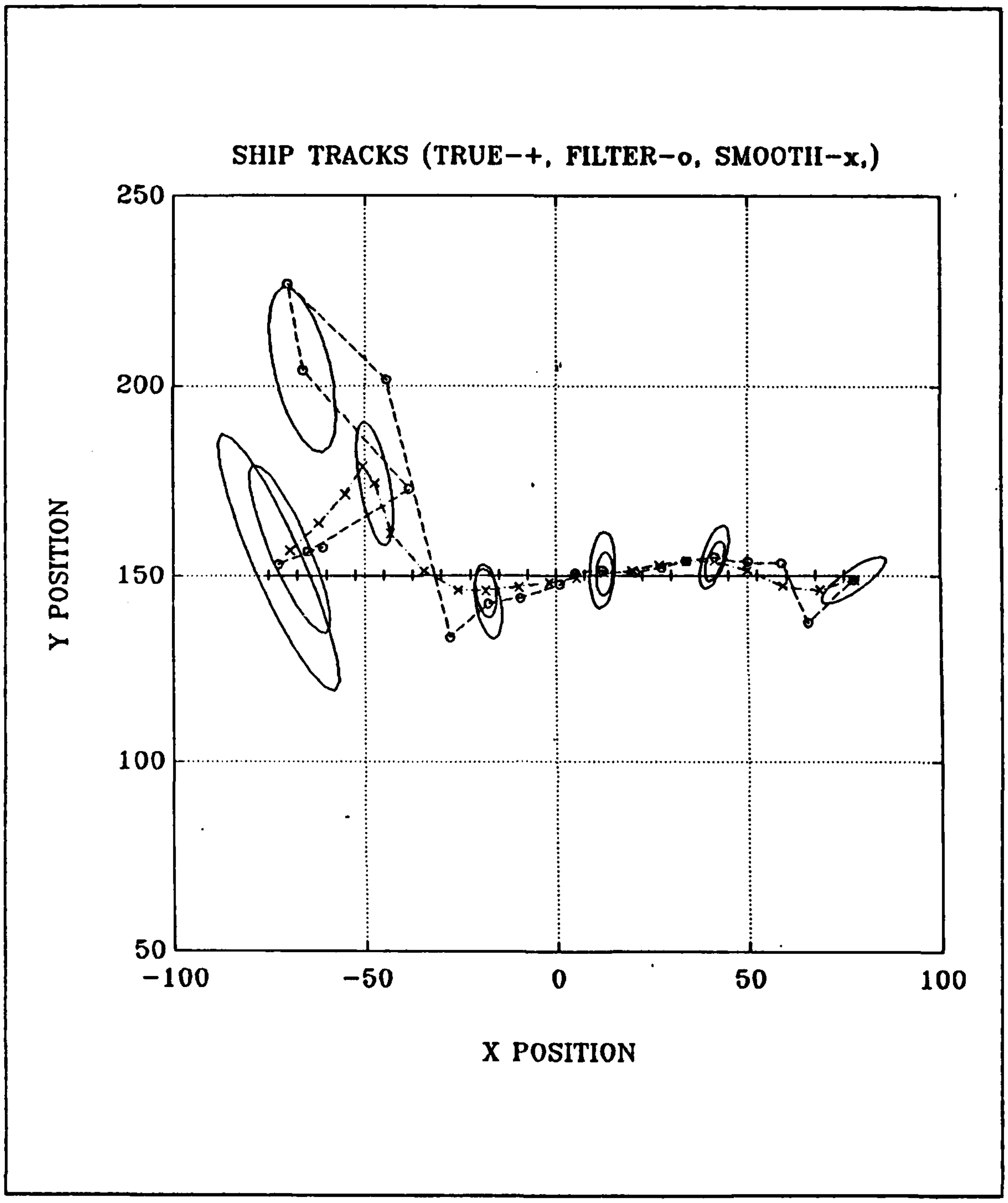

Figure 16. Scenario #3 Overall Track Results

E. SCENARIO #4

In this simulation, the target makes a 60° course change toward the advancing tracking ships five hours after the problem starts. The results for this scenario are shown in Figures 17-20. The atmospheric noise model in this case is white noise. As in Scenario 1, the initial error ellipses indicate the high degree of uncertainty in the position estimates, especially in the *y* direction where the range between the target ship and the tracking ships is the greatest. The average improvement in position error due to the filter for the entire simulation run was 24% and that due to the smoother was 28.5%. However, if we look at the first five hours of the problem, while the ships were still over 100 miles away from each other, the improvement due to the filter was 44.5% and the smoother improved the position accuracy by over 62%. After the target maneuvered, the vessels closed each other at a speed of 23 knots and the position error decreased rapidly. The Kalman filter tracked the target at an average speed of 16.5 knots throughout the problem and the smoothed speed estimate was 14.8 knots.

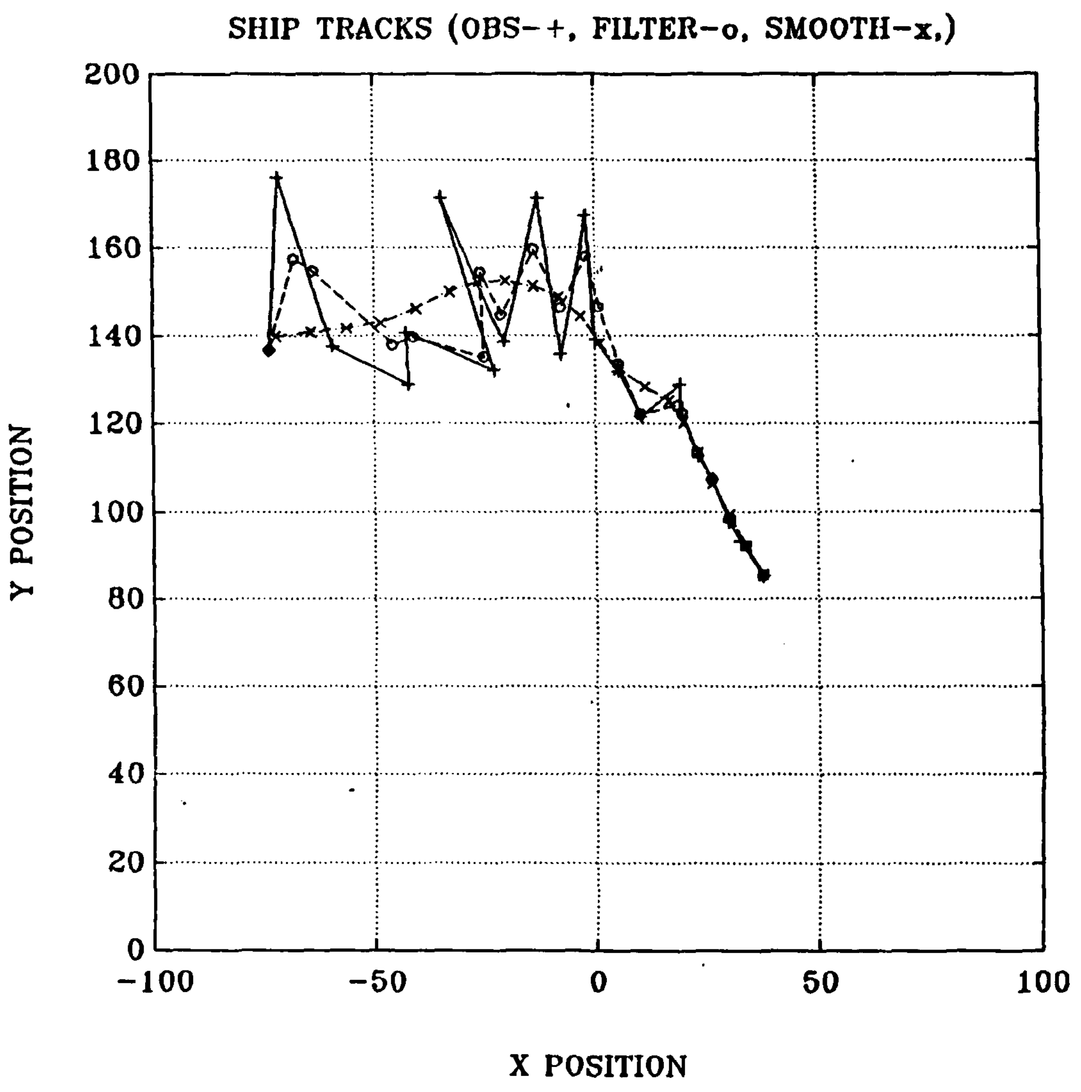

Figure 17. Processed Data vs Observed Results

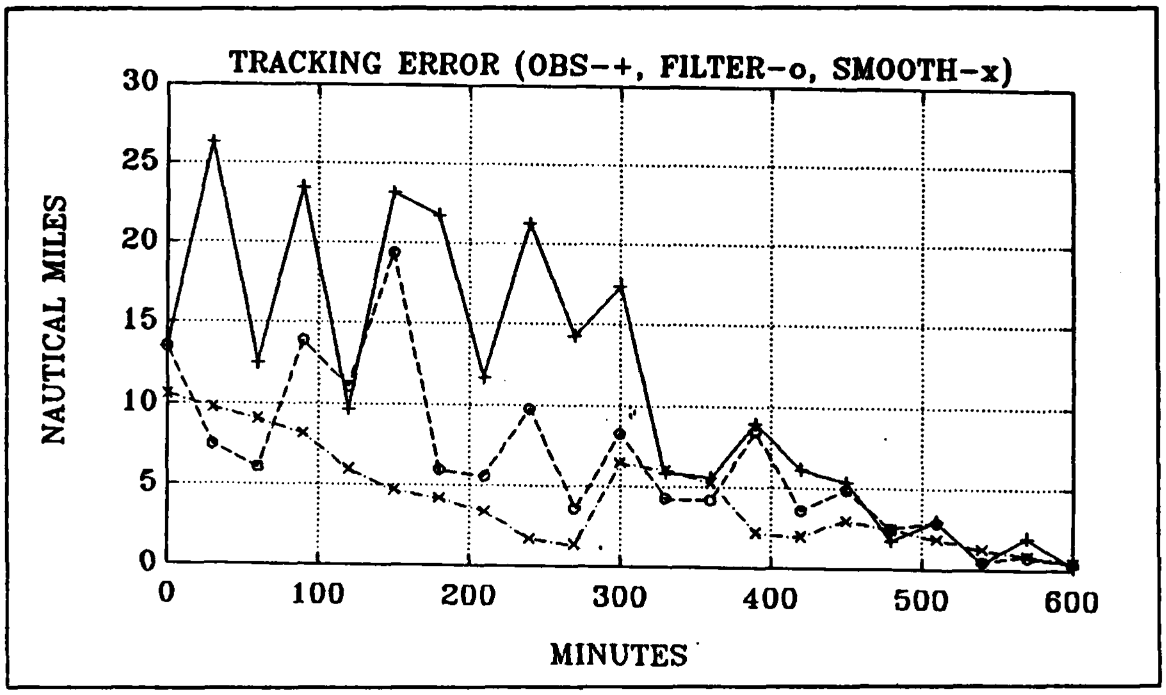

Figure 18. Position Error

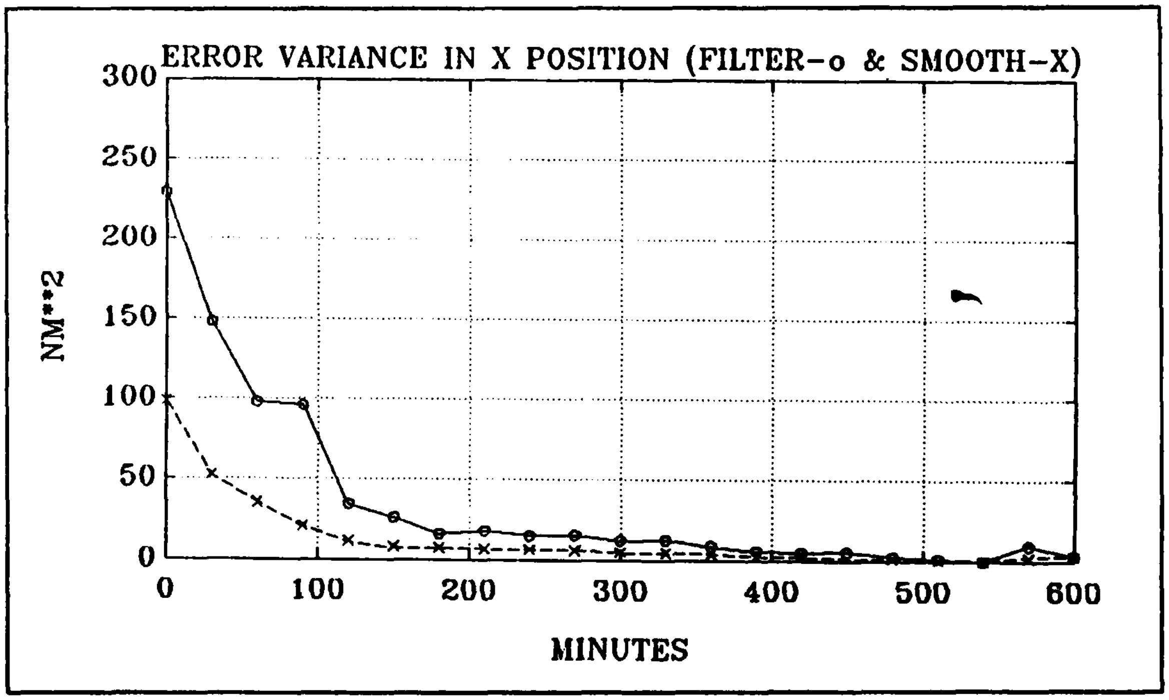

Figure 19. Variance of Position Error in the X Direction

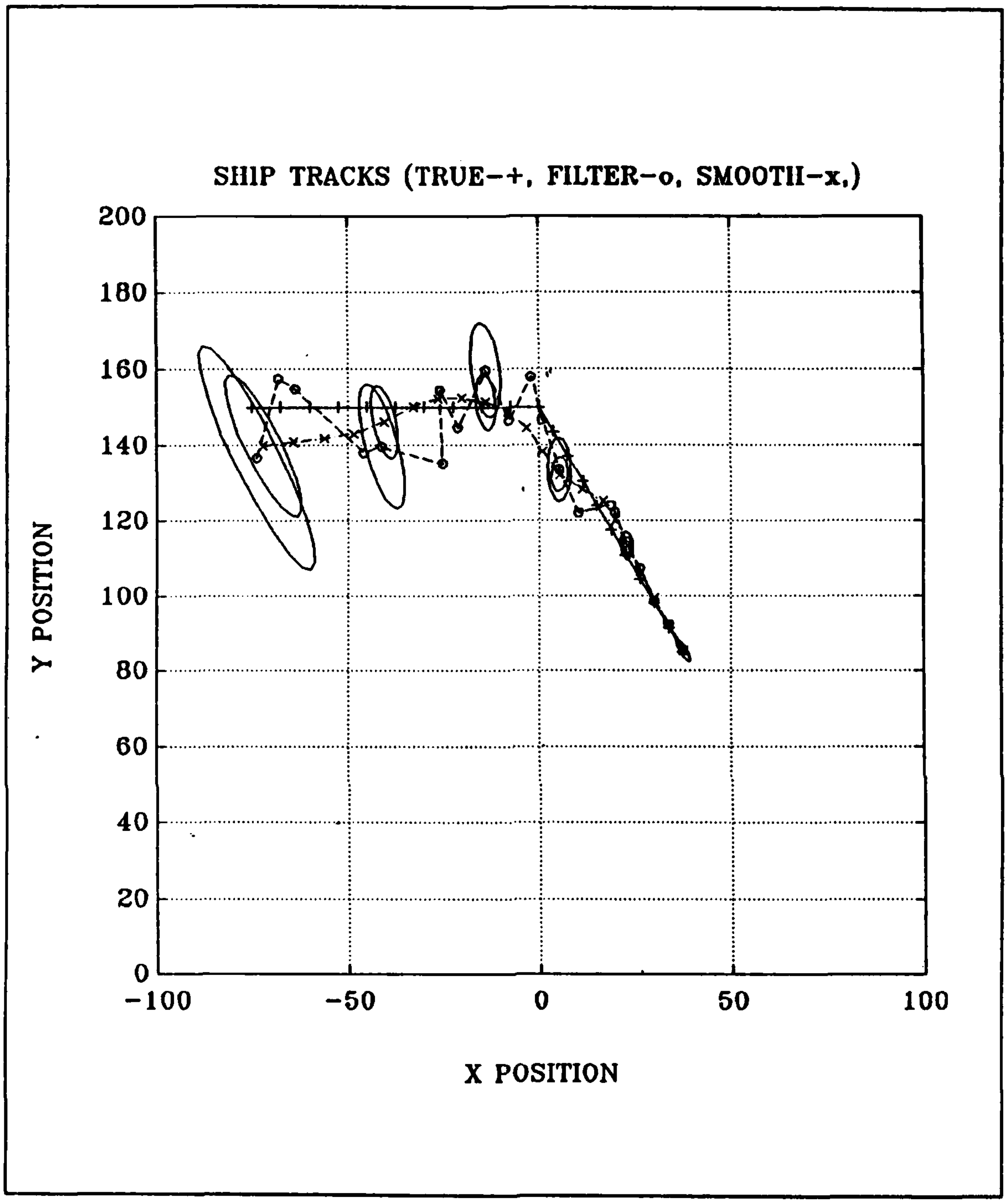

Figure 20. Scenario #4 Overall Track Results

F. SCENARIO #5

The white noise model used in Scenario #4 is replaced by the Hall noise model for Scenario #5. The results for this scenario are shown in Figures 21-24. In this case, three noise spikes occur at observations one, three, and four as seen in Figure 21 and, therefore, the observed positions at these times are extremly inaccurate. The Kalman filter improved the position accuracy by 33% over these four observations while the improvement due to smoothing the estimates was 59%. A fourth noise spike occurred 270 minutes into the problem and although this was not as large as the first three spikes encountered, the filtered estimate is 46% better than the observed estimate and the smoothed estimate is 78% better. Due to the relatively low amplitude of the Hall noise model outside of the noise spikes, the filtered and smoothed position estimates show only a minor improvement in time frame from six to nine hours. The small noise spike at time 540 is pretty well smoothed out as seen in Figure 22. The average speed of the target estimated by the filter is 21 knots and the speed estimated by the smoother is 16.4 knots.

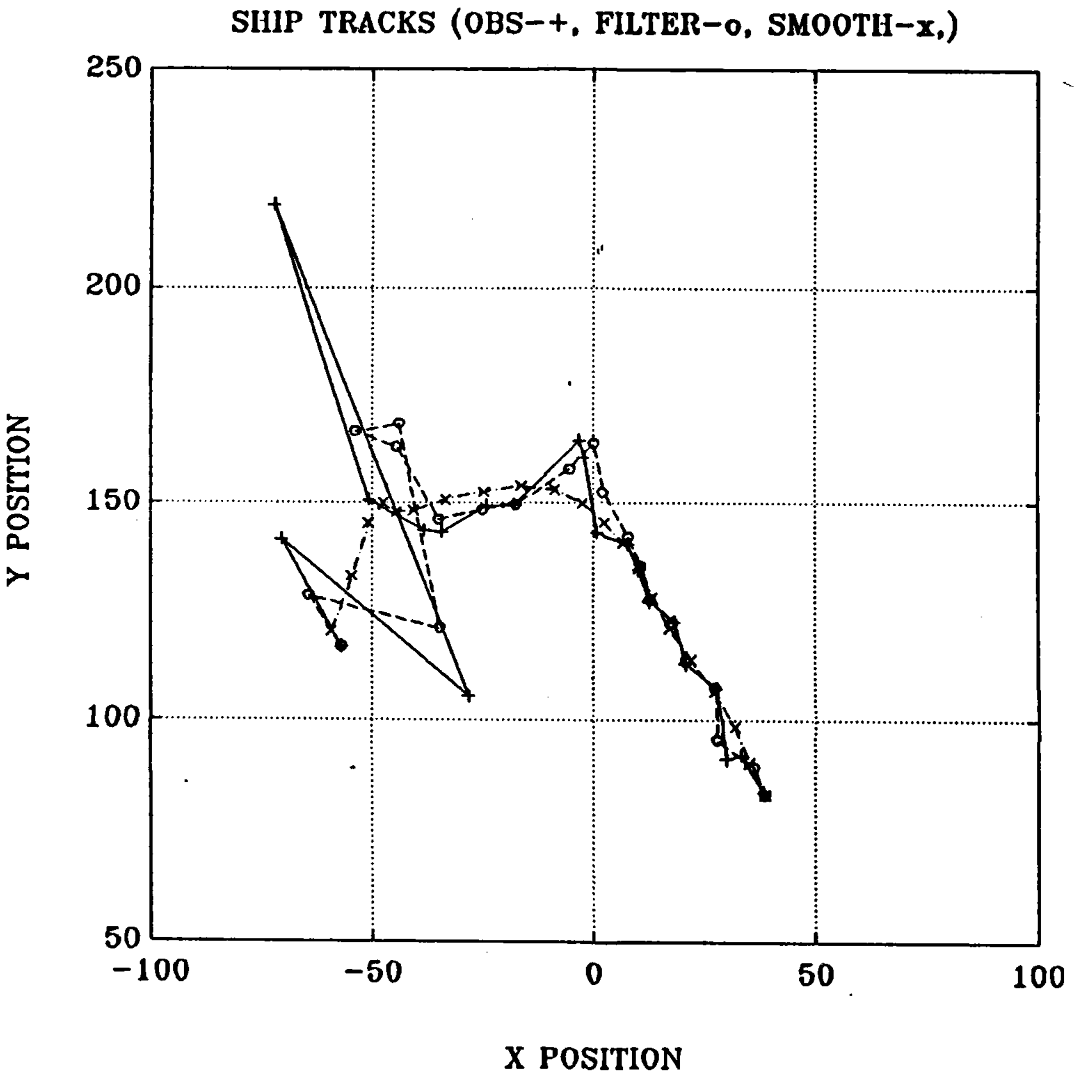

Figure 21. Processed Data vs Observed Results

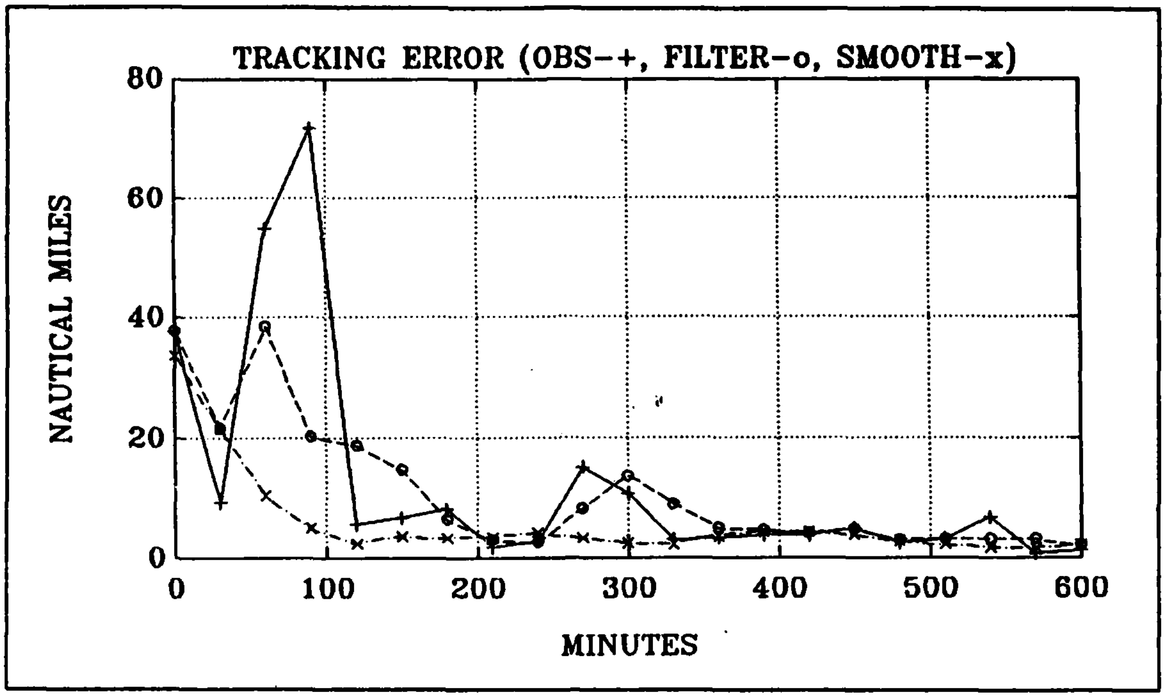

Figure 22. Position Error

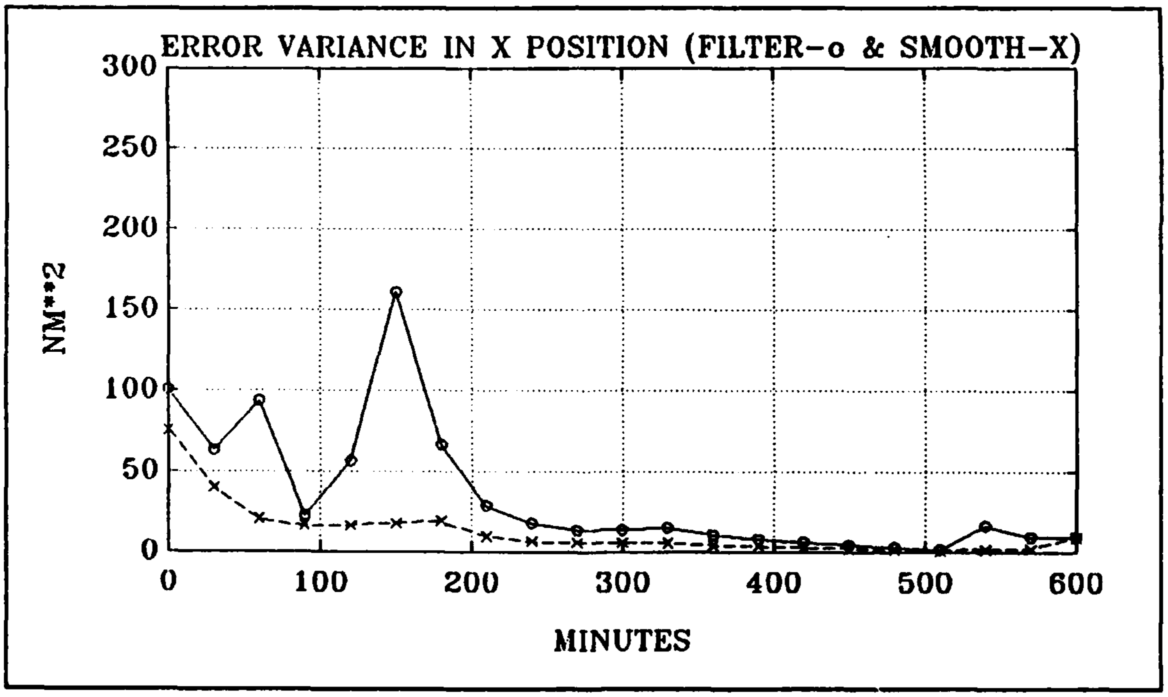

Figure 23. Variance of Position Error in the X Direction

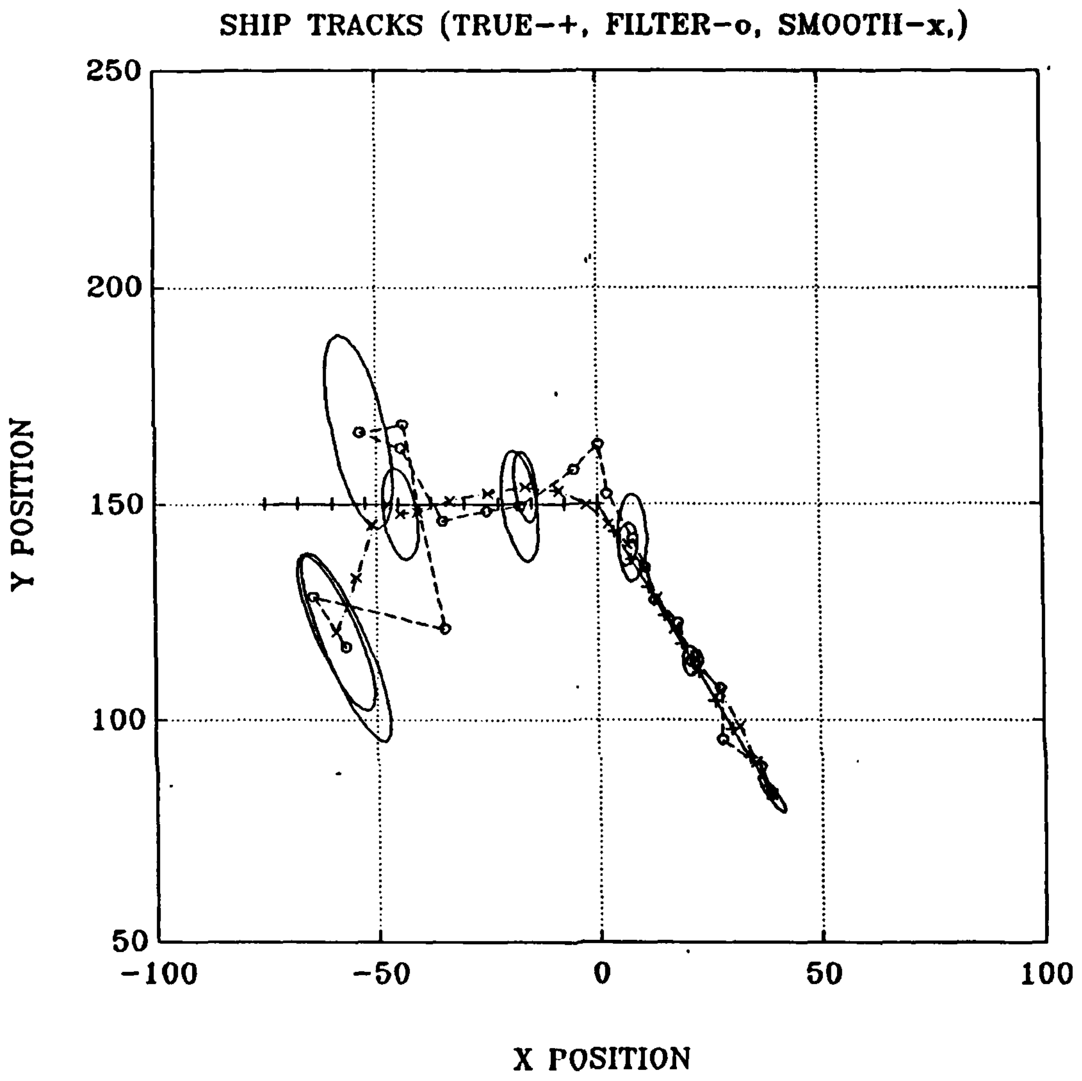

Figure 24. Scenario #5 Overall Track Results

G. SCENARIO #6

This scenario depicts a 60° target maneuver to the northeast away from the two tracking ships heading due north. The results for this scenario are shown in Figures 25-28. The white noise model is used in this simulation and because the relative distance between the ships remains between 100 and 150 nm throughout the problem, the observed position error remains on the order of 10-20 nm. This case demonstrates a general improvement in the filtered and smoothed estimates over the entire track. The position accuracy was increased by 25% with Kalman filter and by 45% using the fixed interval smoothing routine. The smoother kept the track error at or below 8 nm for the entire track interval with the exception of the first three observations to be smoothed at times 600, 570, and 540. The average speed estimates for the filter and the smoother were 17.1 knots and 16.6 knots respectively. Figure 28 shows the orientation of the major axis lying in the north-south direction where the distance between the ships is the largest. The magnitude of this axis is the largest at the beginning and the end of the problem and at a minimum when the target maneuvers and ships are the closest together.

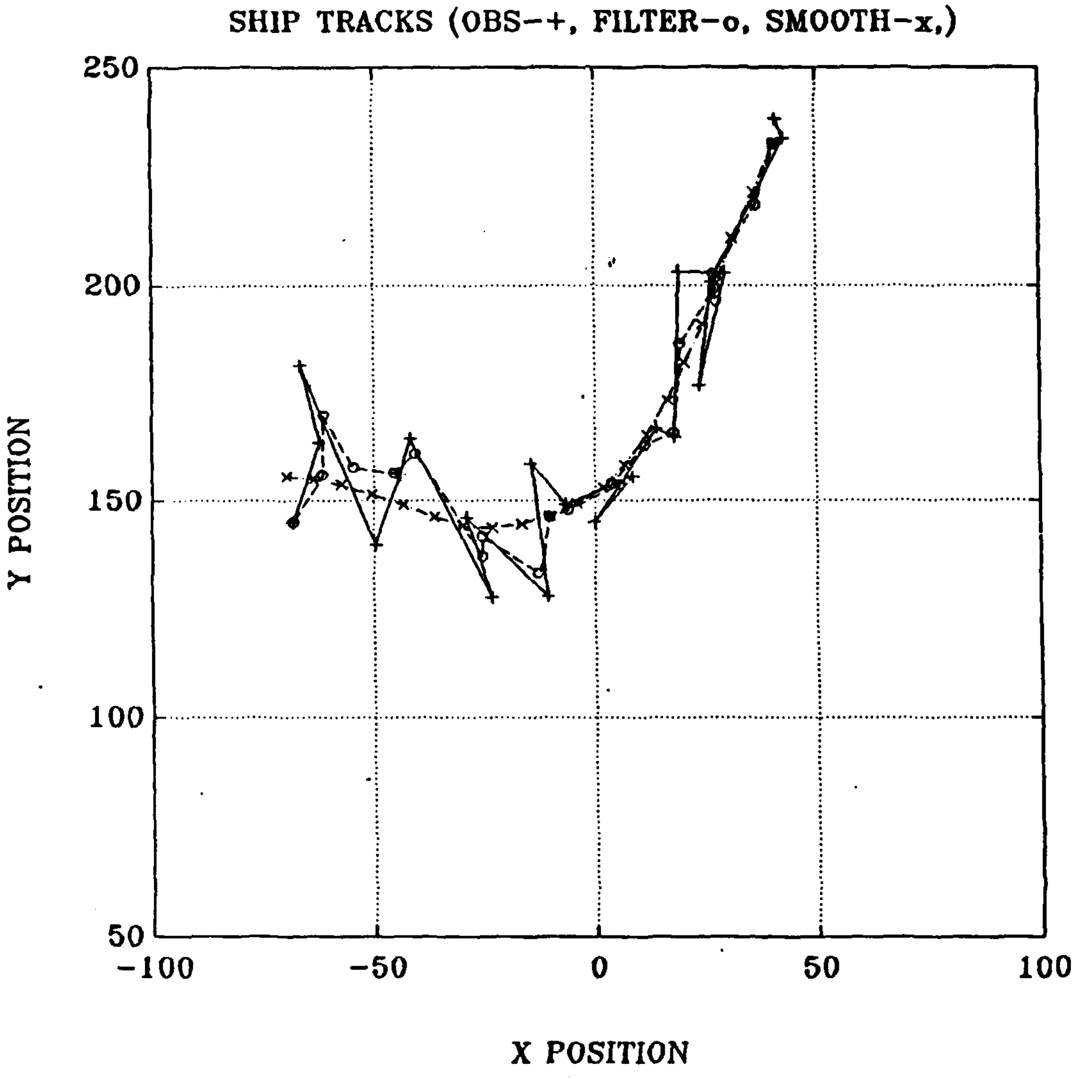

Figure 25. Processed Data vs Observed Results

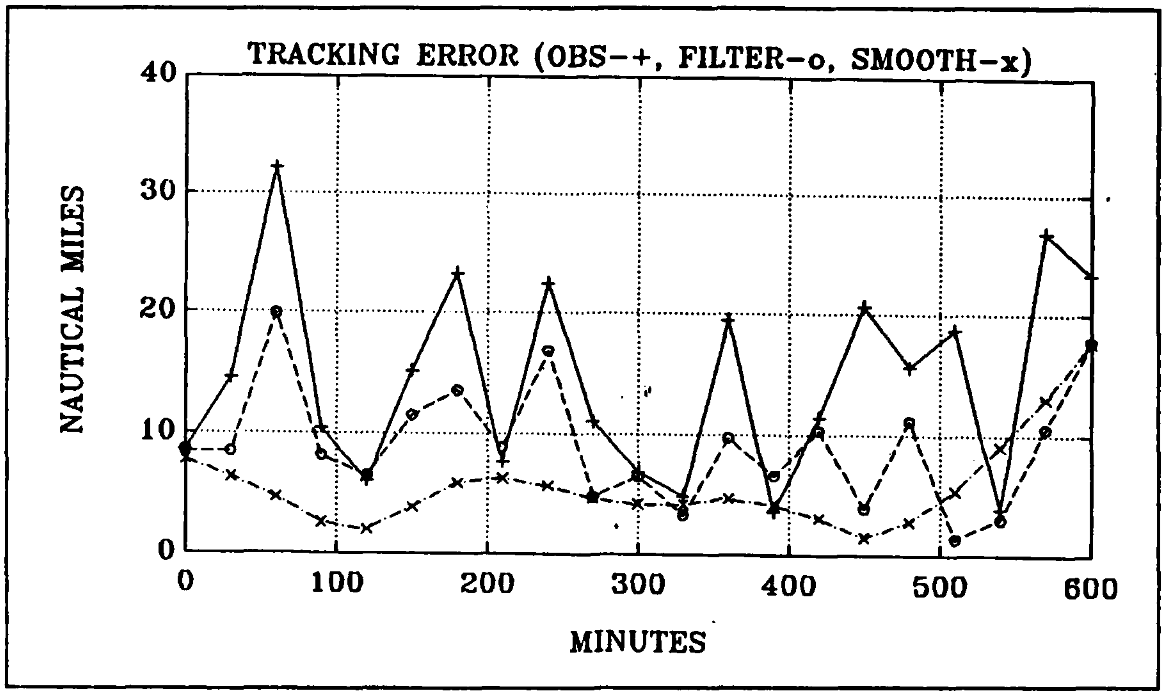

Figure 26. Position Error

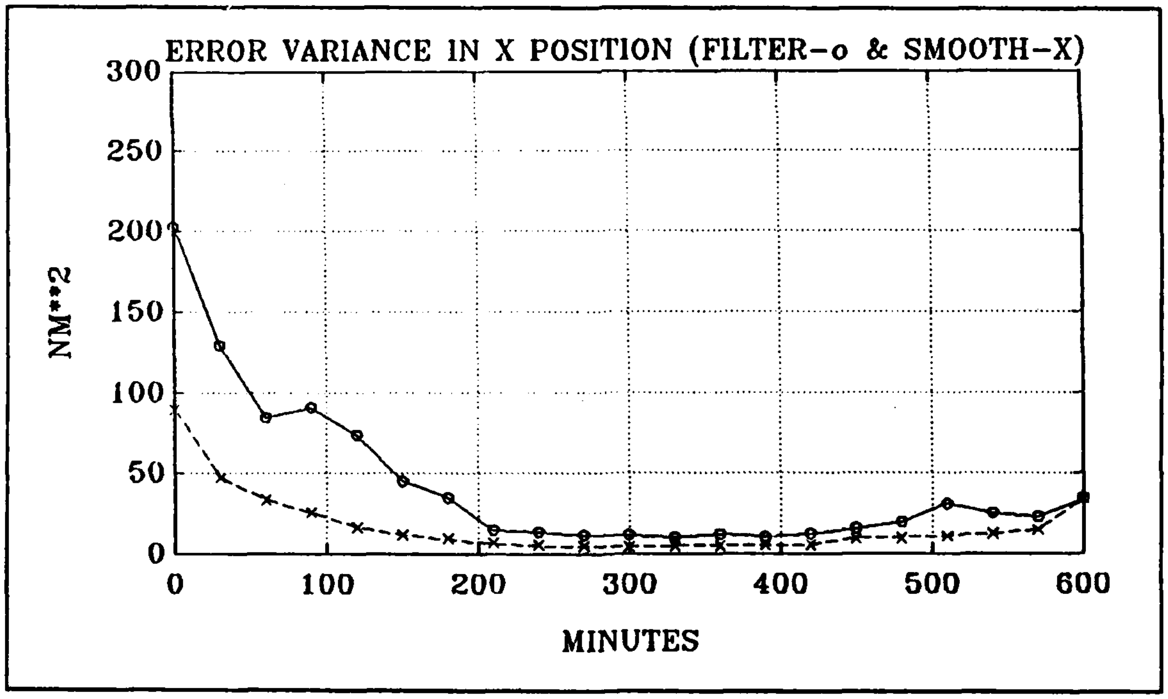

Figure 27. Variance of Position Error in the X Direction

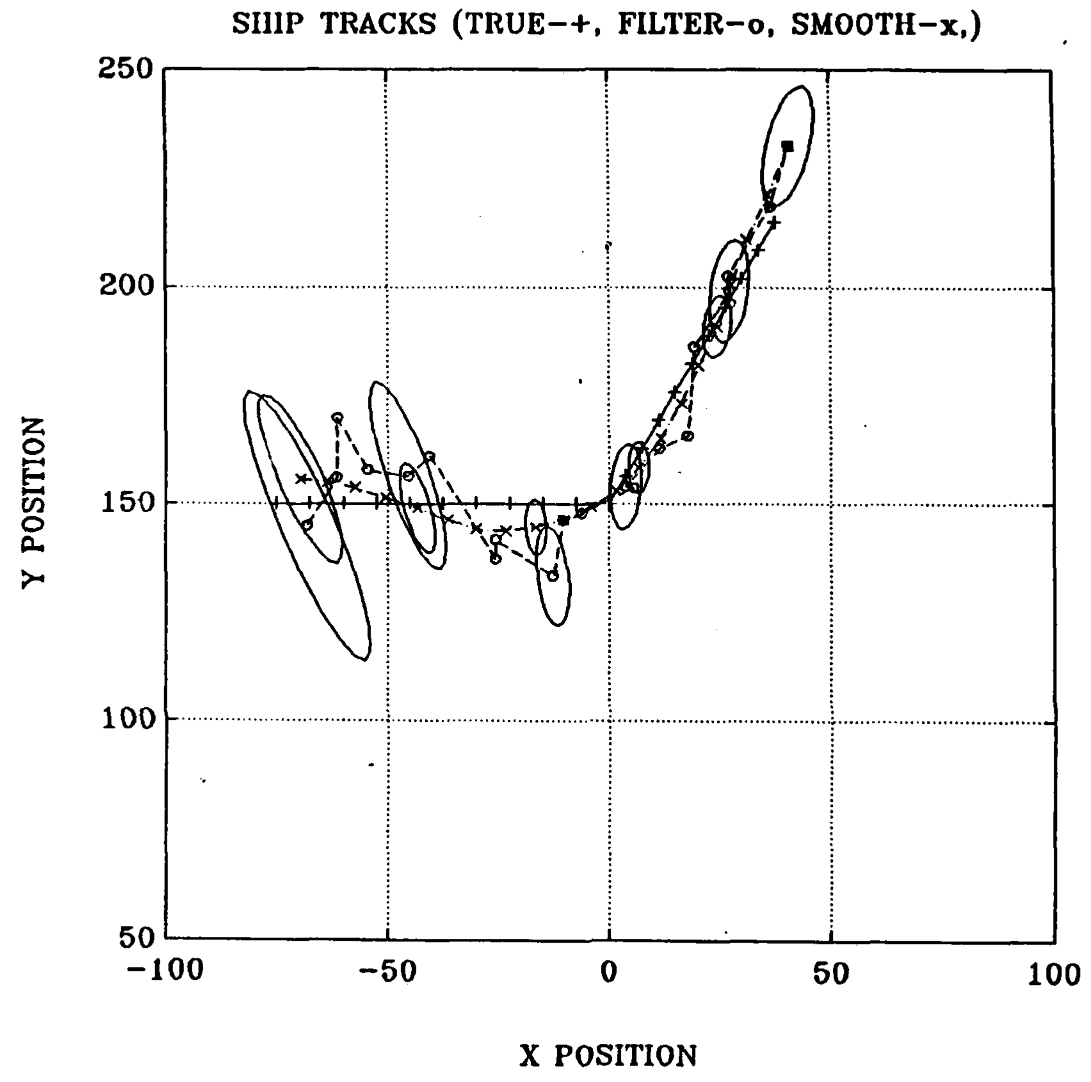

Figure 28. Scenario #6 Overall Track Results

H. SCENARIO #7

In this case, the Hall model was used to model the atmospheric noise. The results for this scenario are shown in Figures 29-32. We can see three distinct noise spikes from the error plot in Figure 30 at 120, 330, and 510 minutes. For each of these times, the filtered estimate is better than the observed estimate and the smoothed estimate is the most accurate of all. At other times, however, the filtered and smoothed estimate are not the most accurate and in fact are sometimes (180, 360, & 450) much worse than the observed error. The advantage to be gained by using the Kalman filter and the smoothing routine in a case like this is the elimination of large, non-predictable errors caused by the impulsive characteristics of atmospheric noise. The smoothed position error remains between 2-11 nm throughout the problem while the filter position error has a range of 2-28 nm and the observed position error varies from 3-38 nm. Figure 32 shows the overall accuracy of the smoothed track compared to the target's true track.

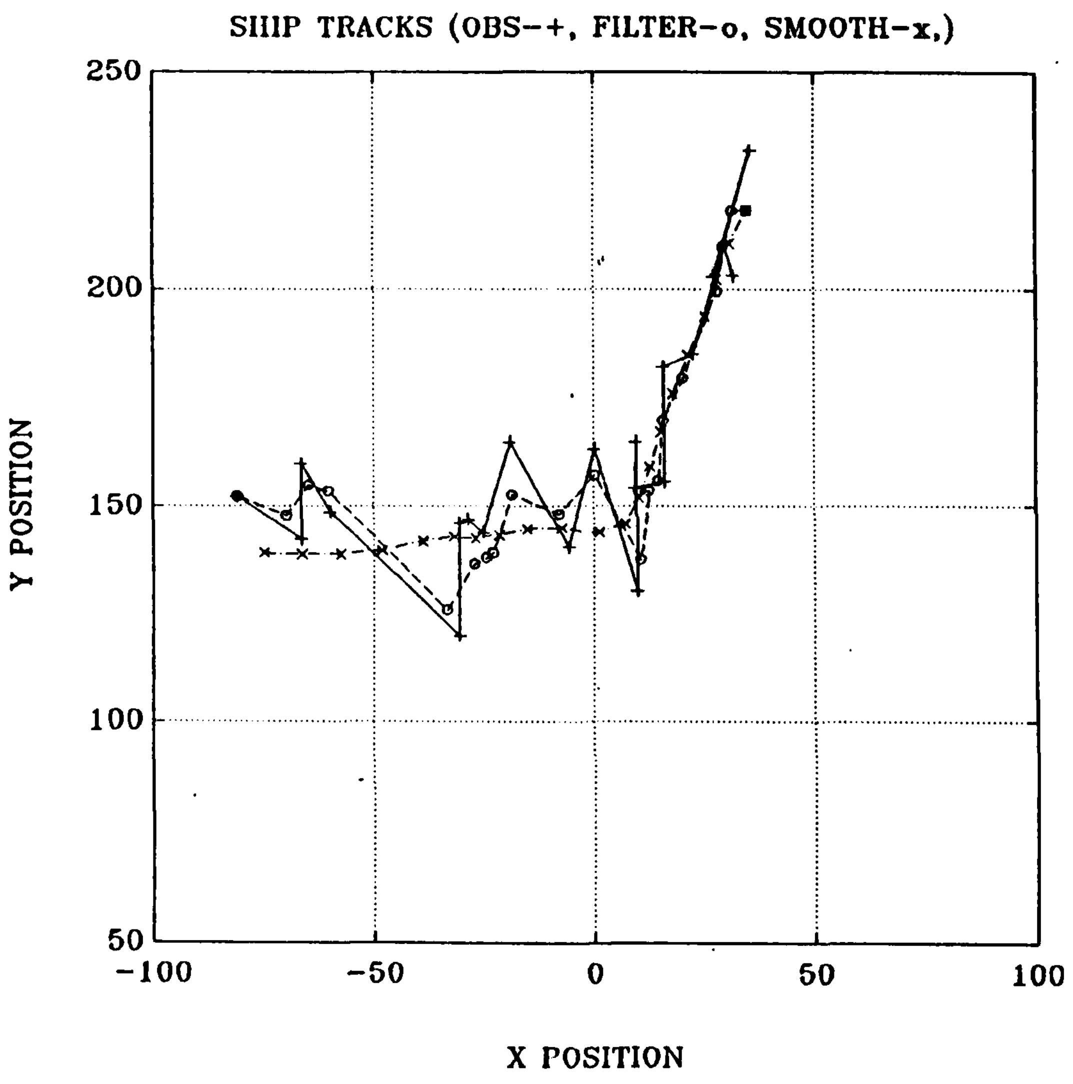

Figure 29. Processed Data vs Observed Results

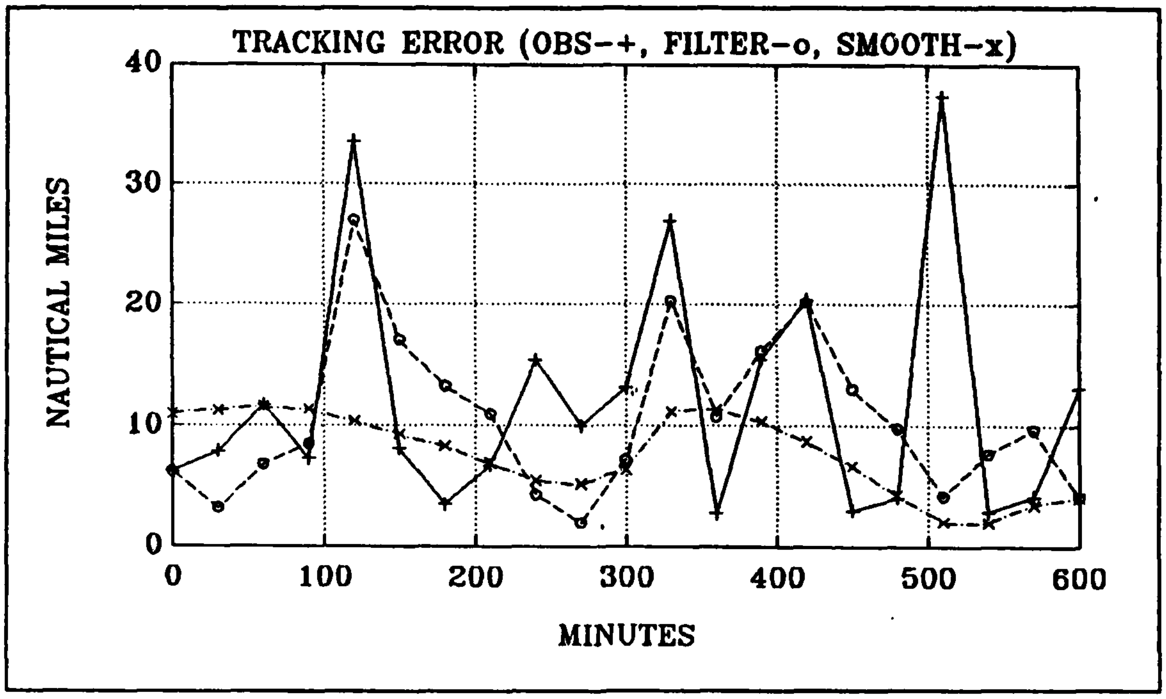

Figure 30. Position Error

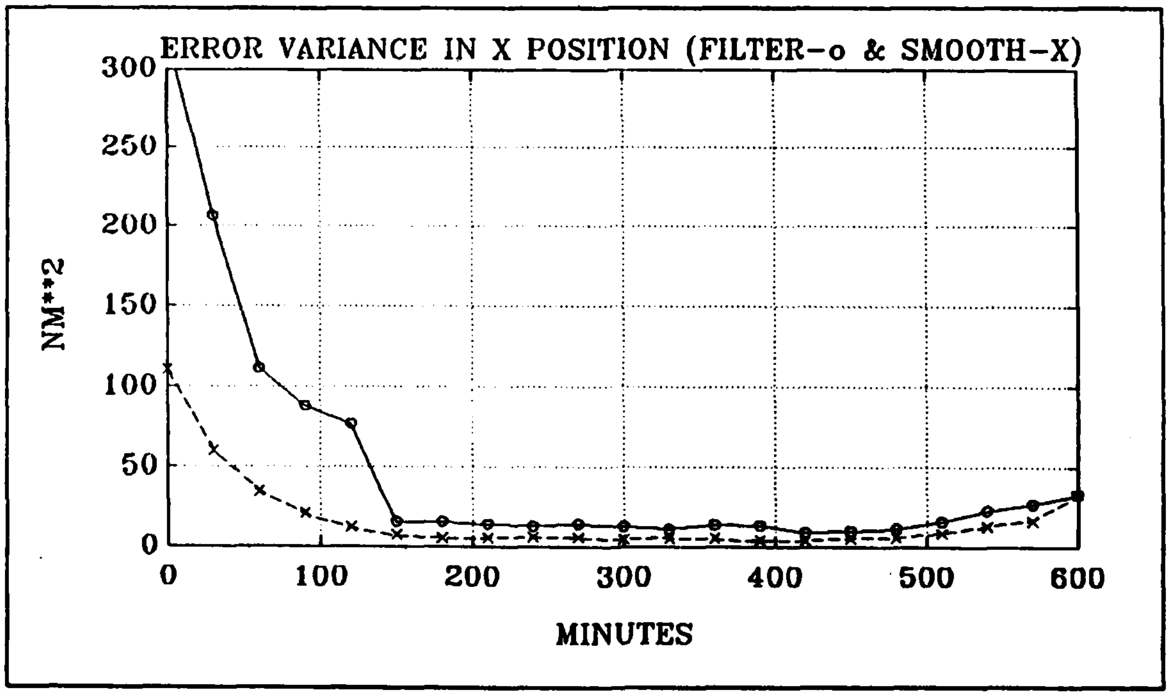

Figure 31. Variance of Position Error in the X Direction

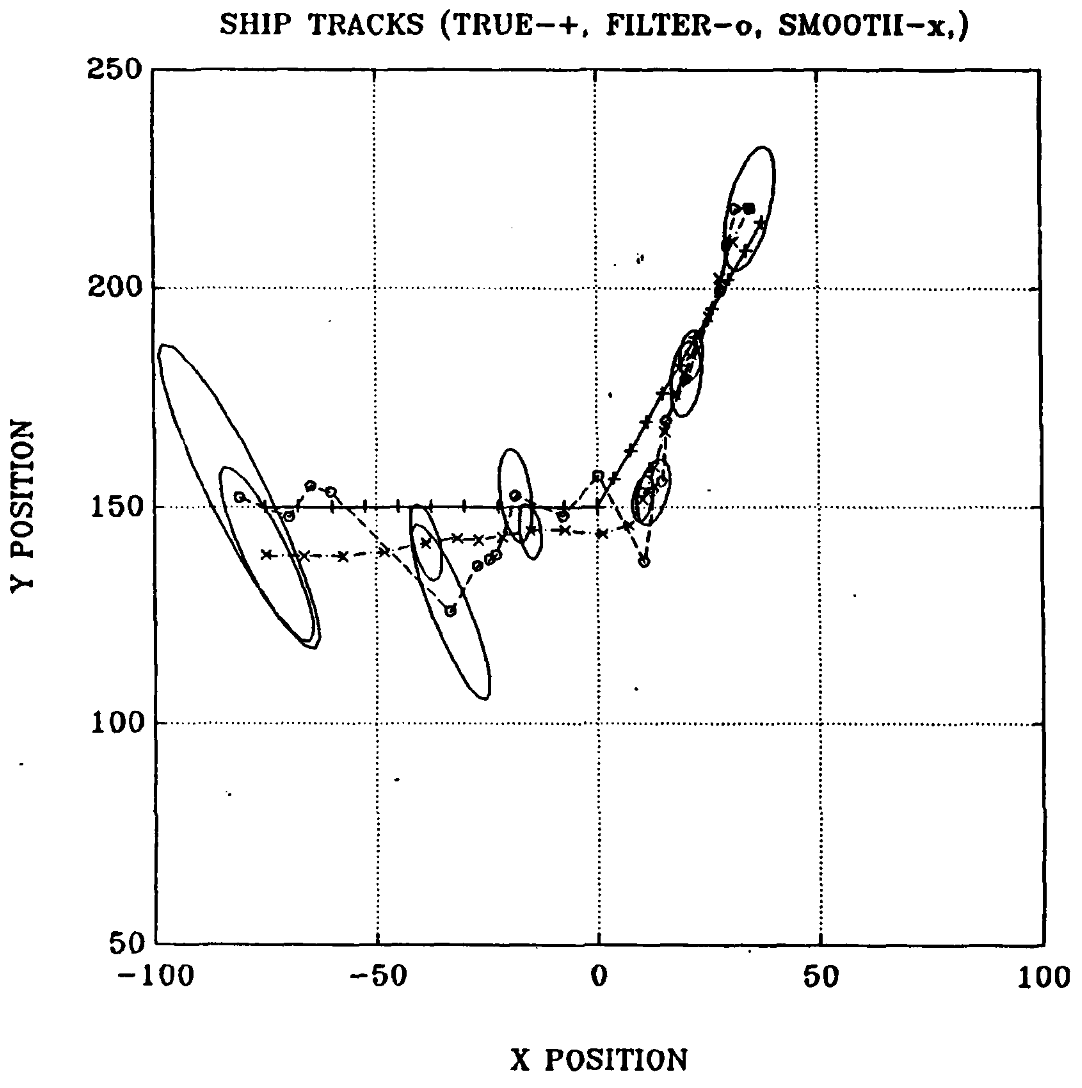

Figure 32. Scenario #7 Overall Track Results

I. SCENARIO #8

This simulation uses the white noise model and includes two target maneuvers during the problem. The results for this scenario are shown in Figures 33-36. The target begins this scenario 150 nm due north and directly between the tracking ships on the *y* axis. The target's initial course is 000° at 15 knots. At time 180 the target turns northwest to a new course 310° . At time 390 the target turns again to a new course 040° . The tracking ships are heading due north at 10 knots. Due to the geometry of this scenario, the tracking routine is relatively accurate in the *x* direction with most of the position error in the *y* direction. The accuracy in the *x* direction is reflected in the small variance error values shown in Figure 35. For the first five hours of the problem the improvement in the position accuracy due to the filter and the smoother was about the same (40%). During the second half of the problem, two large observation errors are detected. In both instances the filter and the smoother improve on the position accuracy, however, the because the observation error at 570 minutes is so large (140 nm), the position errors for the filtered estimates and consequently the smoothed estimates for the last two observations are also very large. The effect of these large bearing errors is that the smoothed target track gives the impression that target has maneuvered and turned north when it actually has not. This can be seen in Figure 36. The size of the major axis of the error ellipses generated by the filtered and smoothed estimates also increases when the estimated distance of the target from the tracking ships increases to more than 220 nm by the end of the problem. The average target speed computed by the Kalman filter is 32.3 knots and the smoothed speed is 23.5 knots.

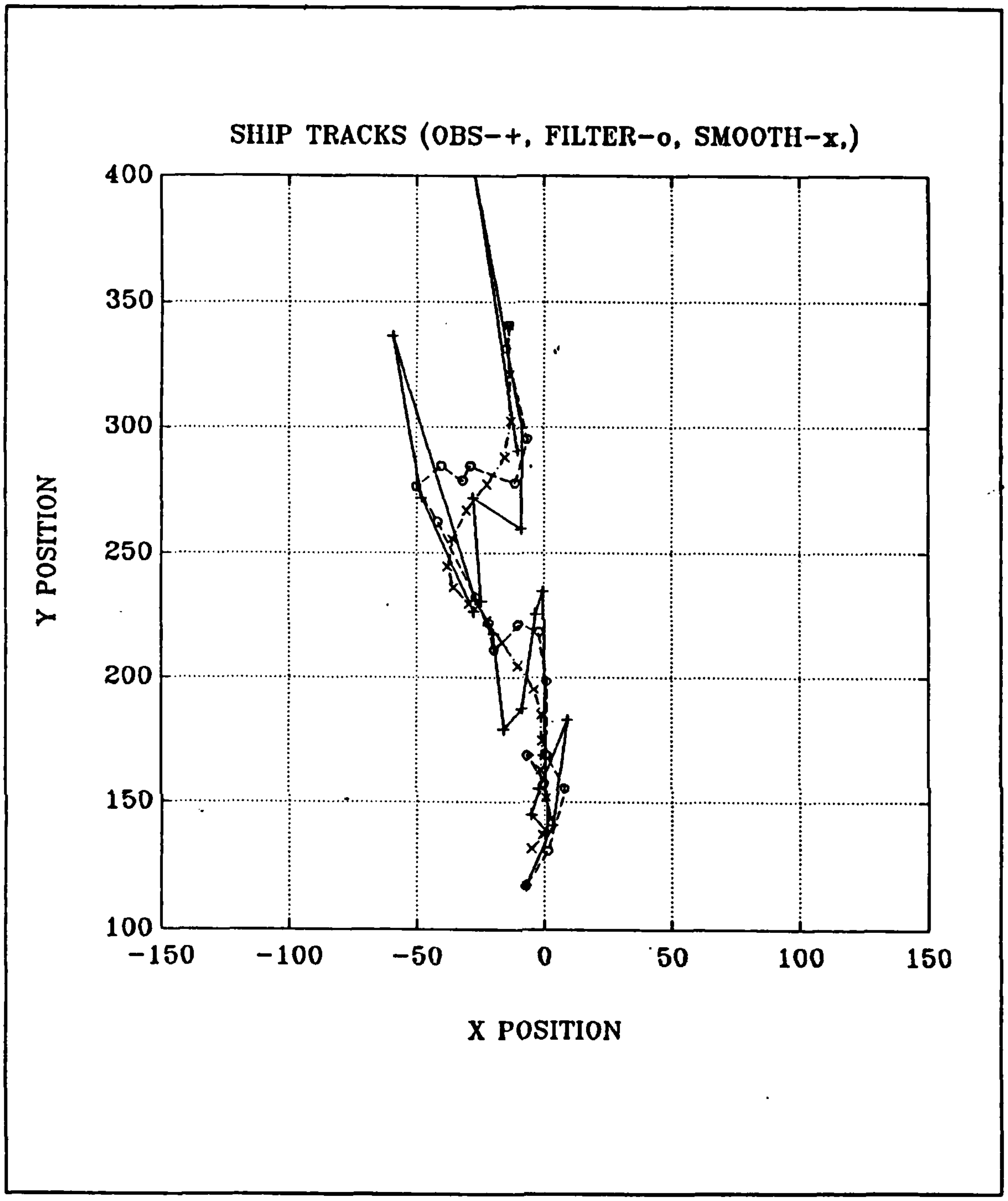

Figure 33. Processed Data vs Observed Results

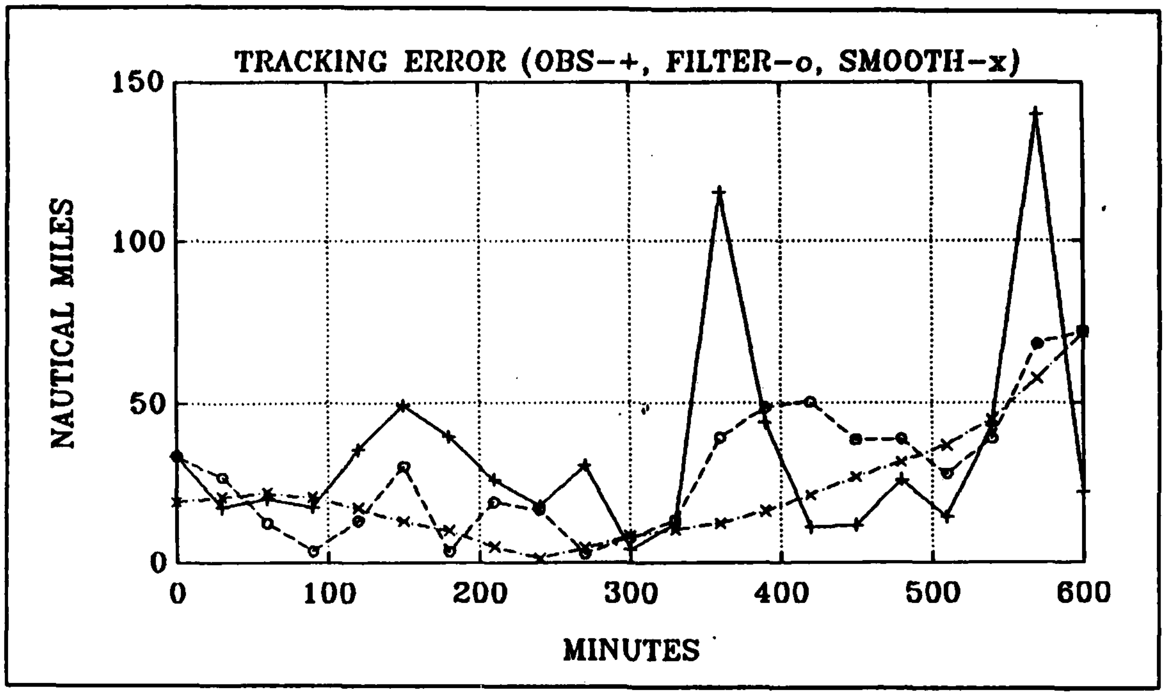

Figure 34. Position Error

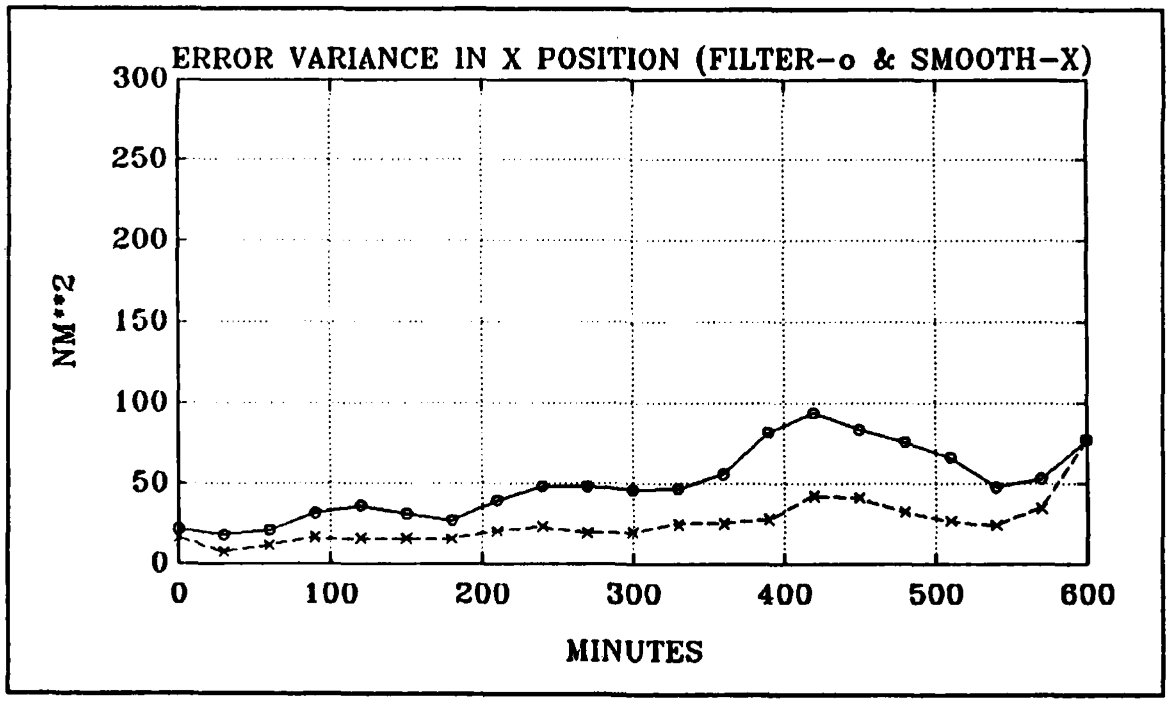

Figure 35. Variance of Position Error in the X Direction

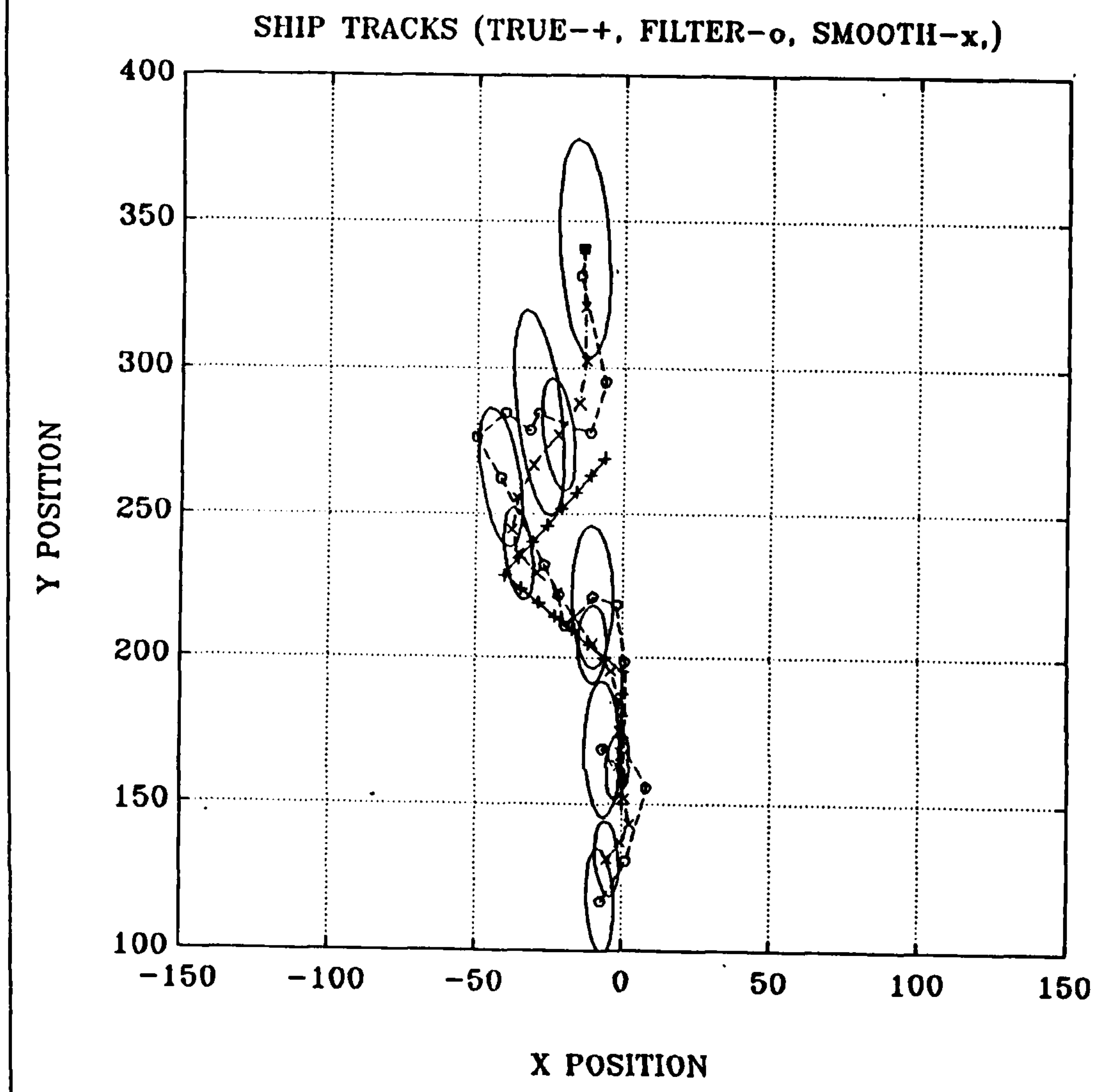

Figure 36. Scenario #8 Overall Track Results

J. SCENARIO #9

Scenario #9 duplicates Scenario #8 with the Hall model used to approximate the atmospheric noise. The results for this scenario are shown in Figures 37-40. Figure 38 shows a large position error due to a noise spike 60 minutes into the problem. The filtered track for this scenario follows the target's true track, however, the position errors vary from 4 to 28 nm over a ten hour tracking period. The smoother improves the position accuracy of the tracker in this problem by an average of more than 60%. The smoothed track follows the target's real track for the entire track period with no major deviations as shown in Figure 40. The average filtered speed is 25 knots and the averaged smoothed speed is 16.5 knots.

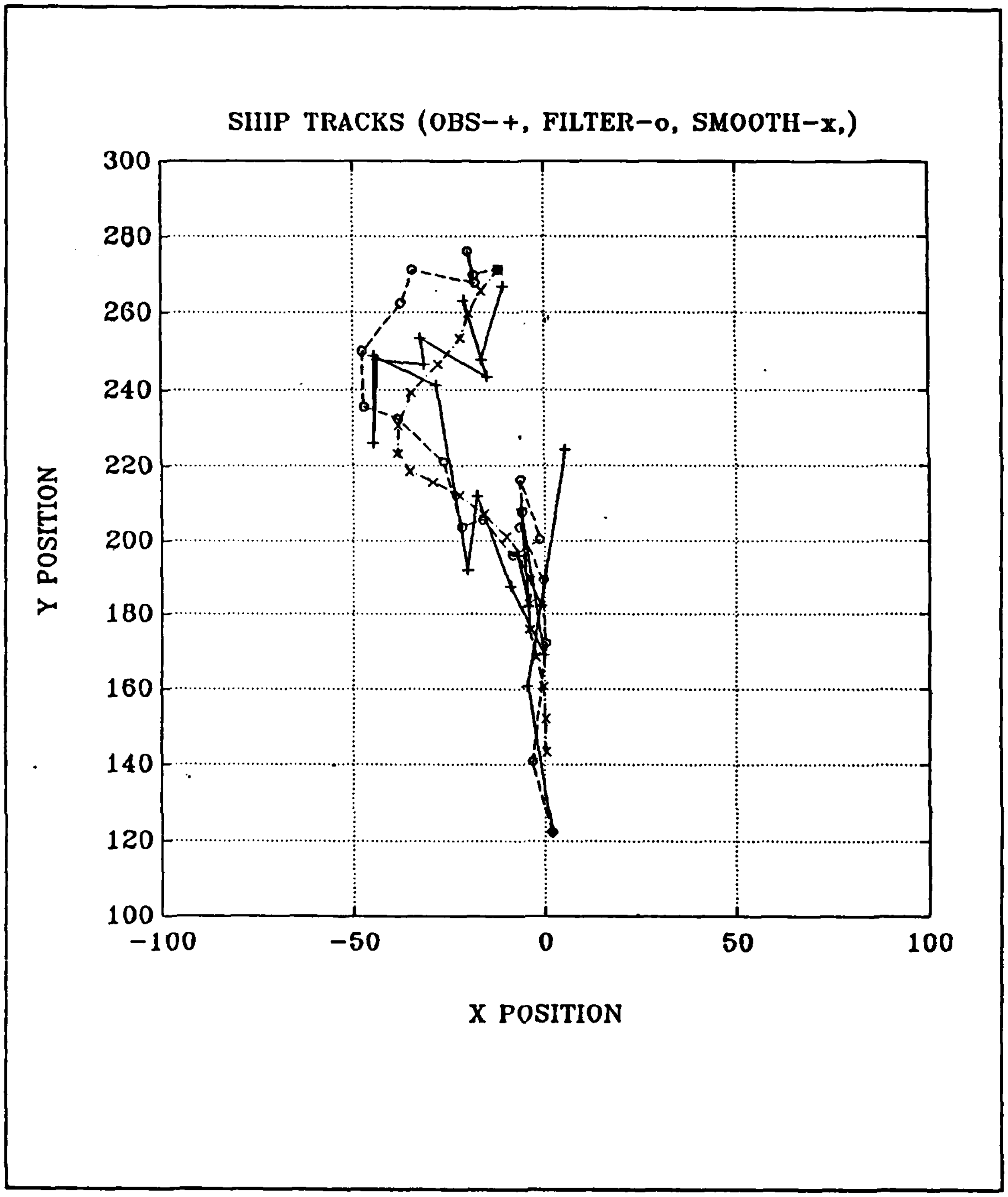

Figure 37. Processed Data vs Observed Results

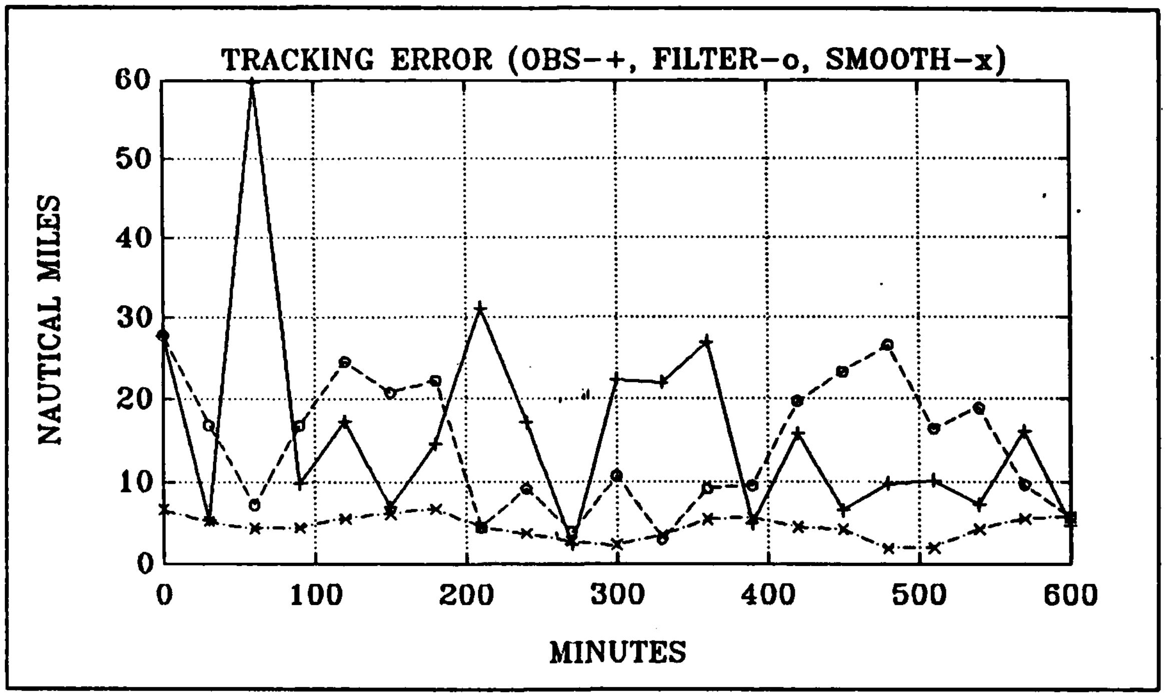

Figure 38. Position Error

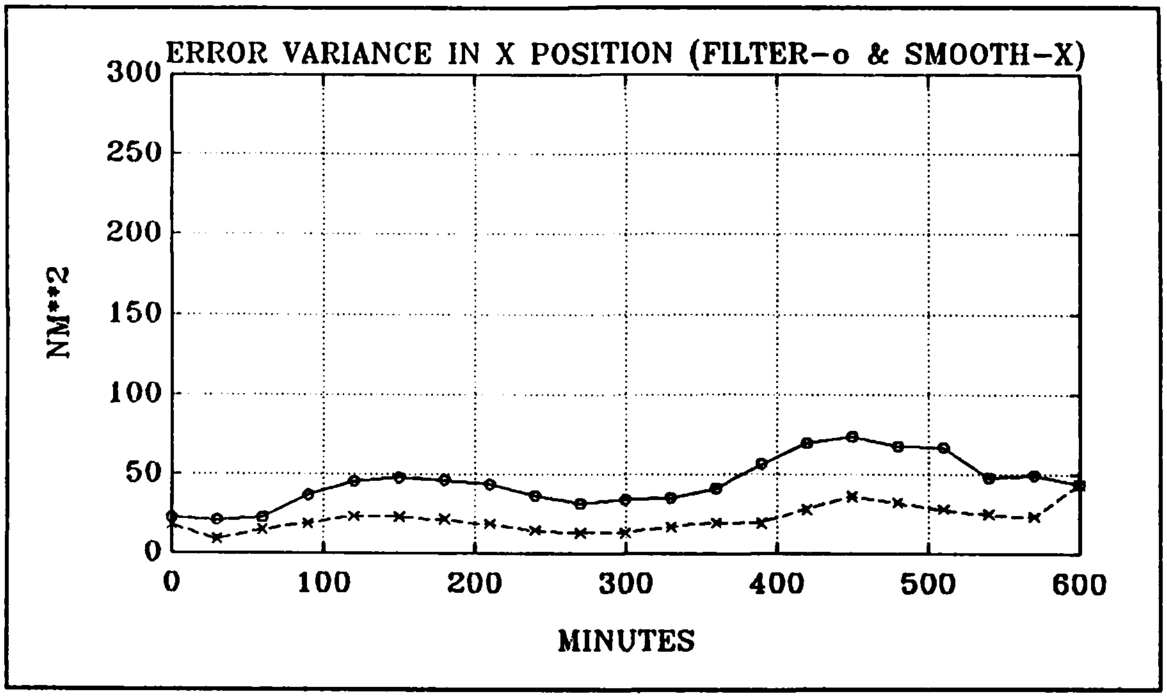

Figure 39. Variance of Position Error in the X Direction

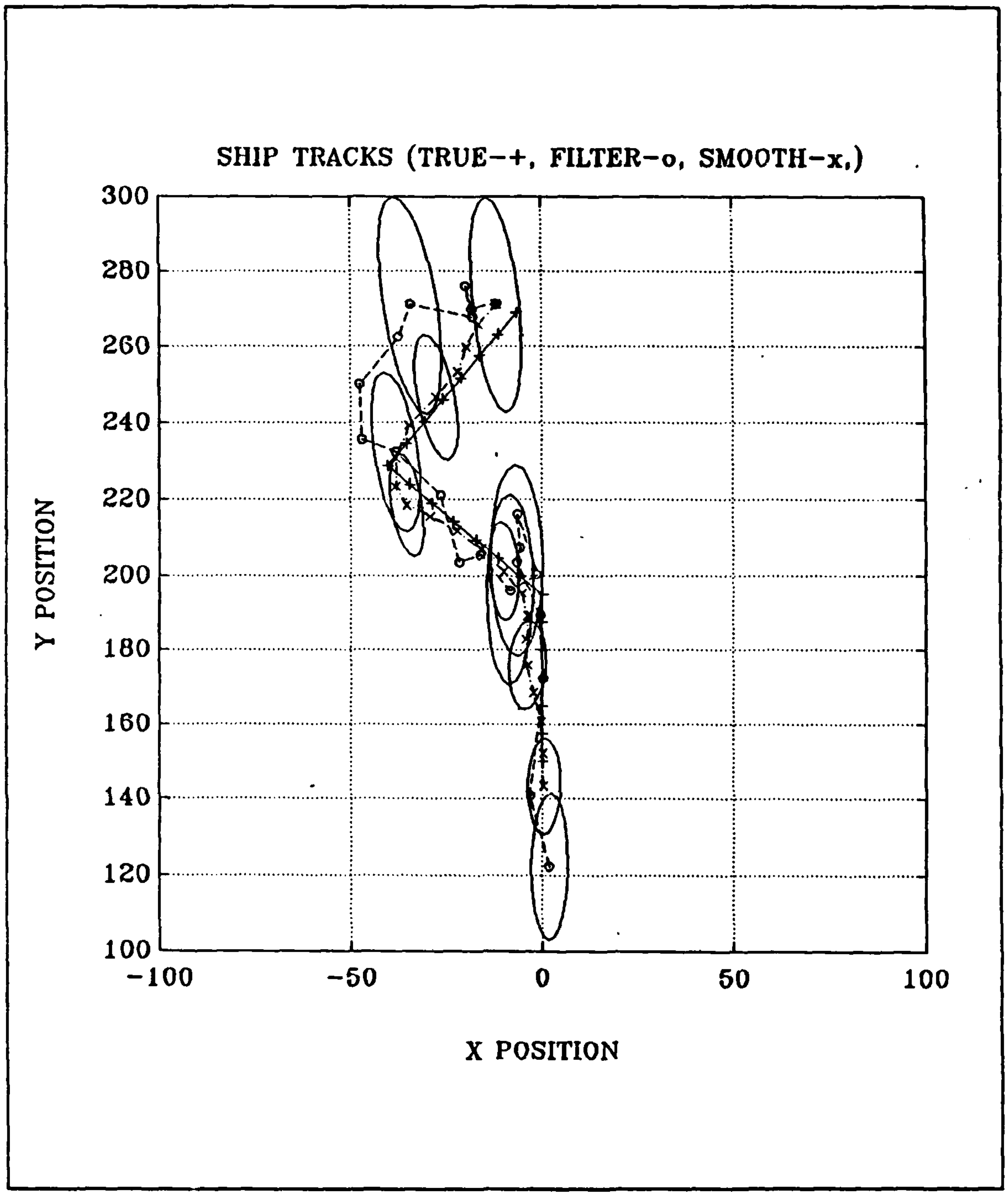

Figure 40. Scenario #9 Overall Track Results

V. STORM TRACKING

A. GENERAL

The storm tracking scenario parallels the ship tracking problem in that both problems develop a position, course, and speed solution for a "target" with similar system dynamics. Where the observations for the ship tracking problem were RDF lines-of-bearing that resulted in a nonlinear measurement equation (2.5), the observations for the storm tracking scenario are actual position coordinates given by latitude and longitude values. The storm tracks used in this thesis were obtained from data collected at the Joint Typhoon Warning Center located in Guam and run by the U.S. Navy and the U.S. Air Force. The position coordinates were obtained using aircraft, satellite, and radar reconnaissance assets.

B. SYSTEM AND MEASUREMENT MODELS

The tropical storms were modeled as linear, time distance systems where the state relationships are given by

$$\begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}_{k} + w_k \tag{5.1}$$

where w_k is a random forcing function with a covariance matrix given by the state excitation matrix, Q_k. This is basically a fictitious noise source that prevents the error covariance matrix from becoming singular. Q_k is defined as

$$Q_k = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix} \tag{5.2}$$

The measurement model utilizes the linear measurement equation given by equation (2.6). Since the x and y position states are observed directly and given by the latitude and longitude position coordinates, the measurement equation can be written as

$$\begin{bmatrix} z_x \\ z_y \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}_k + v_k \qquad (5.3)$$

where the measurement noise v_k has a variance of $\pm$ 1 nm.

C. STORM TRACKS

Typhoon Pat developed east of Taiwan in the western Pacific on 24 August 1985. The Typhoon Warning Period for this storm was from 27 August until 1 September. Over this six-day period the storm traveled 1337 nm with maximum wind speeds of over 95 knots. This storm caused significant damage in southwestern and northeastern Japan; primarily on the islands of Kyushu and Hokkaido. [Ref. 11: pp. 64-68]

Figure 41 shows the actual track of Typhoon Pat using the observed positions.The storm track shown in Figure 42 was constructed using the filtered and smoothed position estimates obtained from the Kalman tracker. The observed storm positions are fairly accurate and do not deviate from the overall track very much. Consequently, the filtered and smoothed storm positions do not show a great deal of improvement overall. In the areas of the track where the observed positions do vary, the smoother does improve the track accuracy. Specifically, these areas occur near 22° N, 126° E, and 37° N, 133° E.

The second storm track analyzed was that of Typhoon Tess that originated southeast of Guam on 30 August 1985. The warning period for this storm lasted from 1 September until 6 September. Tess traveled 1470 nm over its' seven-day lifespan on a track that took it over the Philippines, north of Luzon, and across the South China Sea moving inland 170 nm south of Hong Kong. Maximum wind speeds for this typhoon were over 75 knots. [Ref. 11: pp. 76-77]

The performance of the smoother on the track of Typhoon Tess was similar to that of Typhoon Pat. Figure 43 shows Tess's overall track and Figure 44 shows the track results obtained with the Kalman filter and the smoothing algorithm. The Kalman tracker and the smoother show some improvement in the track accuracy in the area of 15° N, 123° E and 18.5° N, 117° E.

The application of the Kalman filter tracker to the storm tracking problem would be very useful in attempting to predict the storm's track when all the position data is not available and the data that is available is not refined. Then, by using the filter and smoothing routine, a more accurate track of the storm's past history can be calculated allowing for a more accurate prediction of the storm's future track.

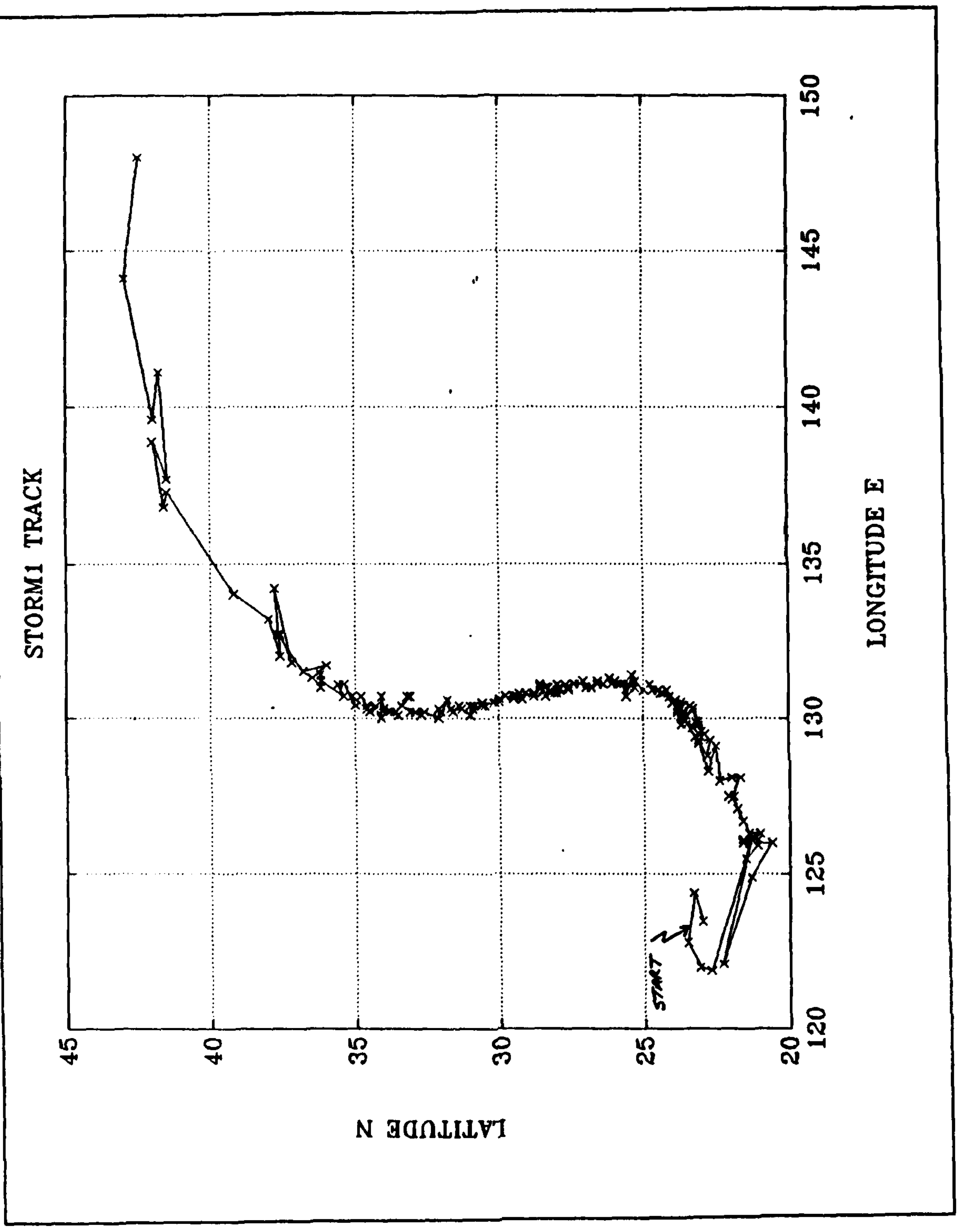

Figure 41. Storm Track of Typhoon Pat

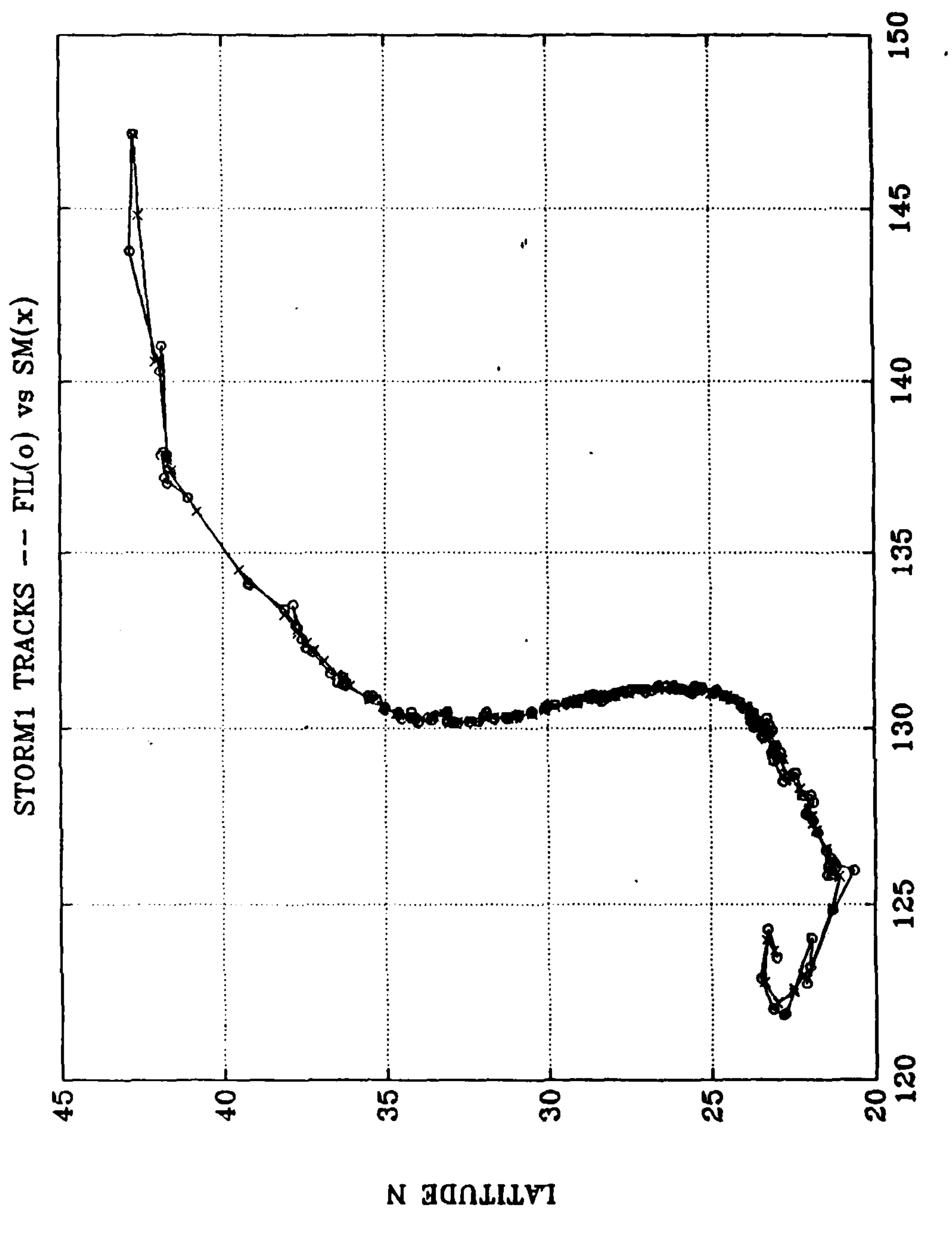

Figure 42. Filtered and Smoothed Track of Typhoon Pat

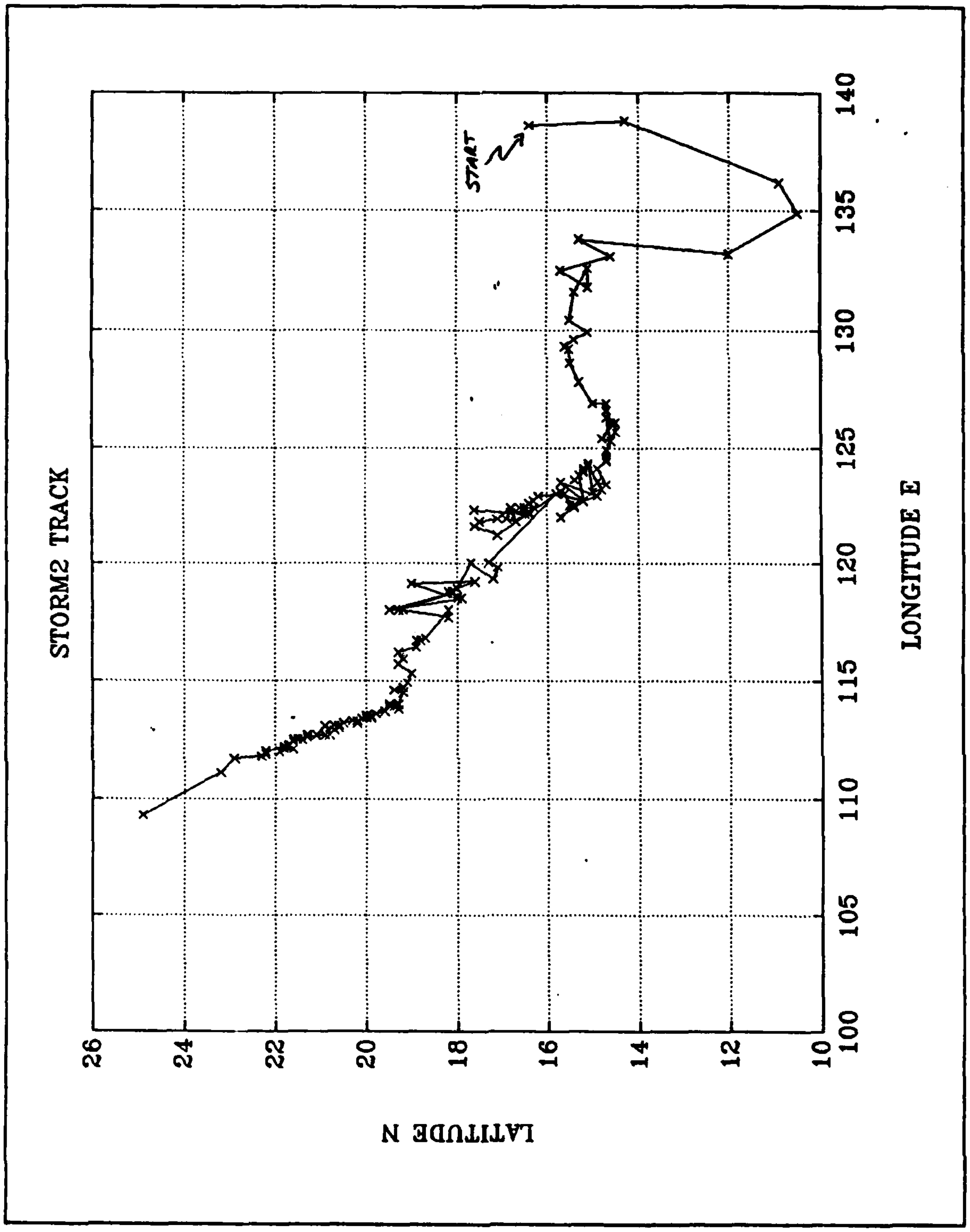

Figure 43. Storm Track of Typhoon Tess

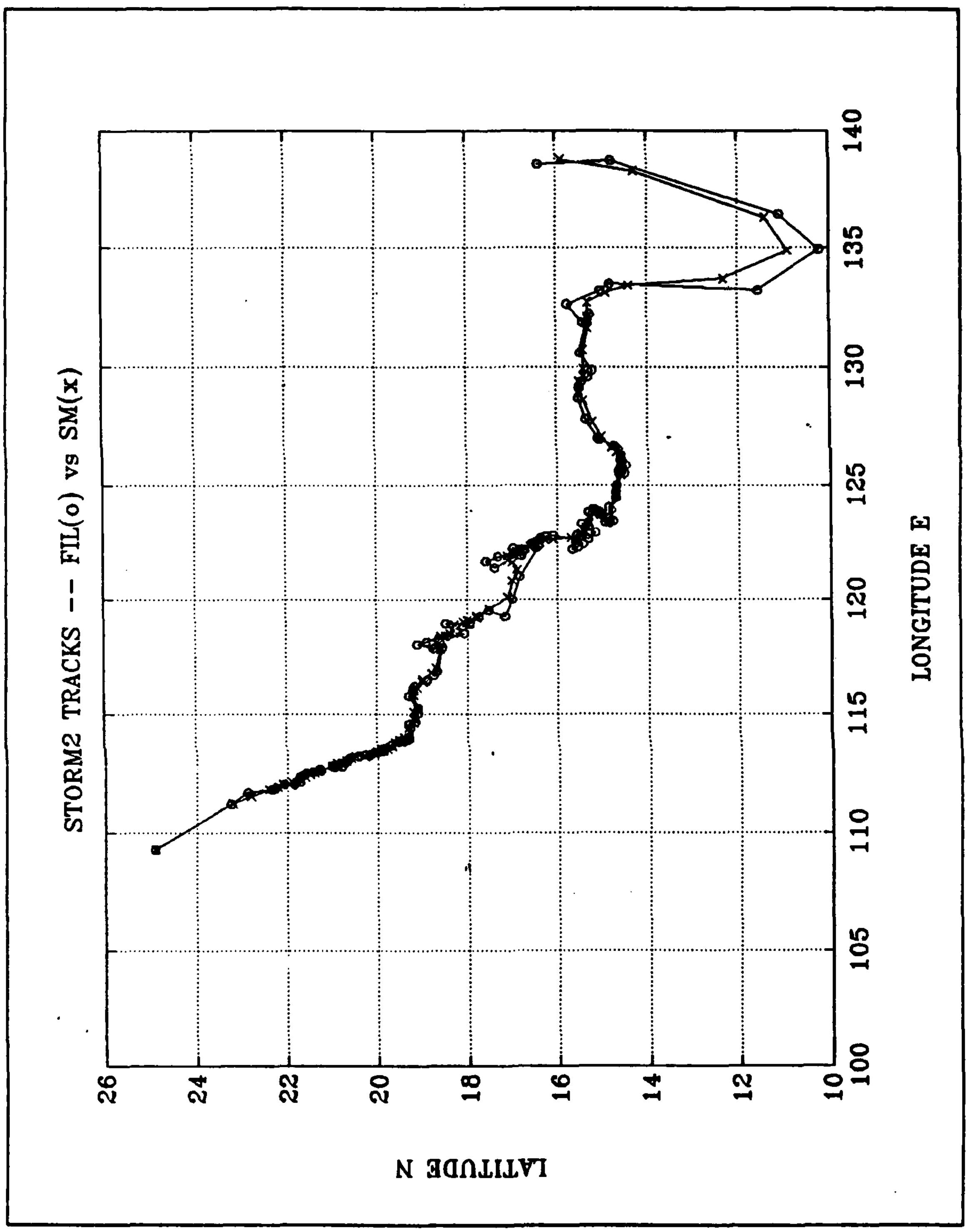

Figure 44. Filtered and Smoothed Track of Typhoon Tess

VI. CONCLUSIONS

The purpose of this research was to improve the accuracy and maneuver detection capability of an extended Kalman filter tracking routine by implementing a fixed interval smoothing algorithm and a maneuver/divergence detection method that uses a noise variance estimator process. The Hall atmospheric noise model was compared with a white noise model in estimating the effects of atmospheric noise on the received HFDF lines-of-bearing. Several different targeting scenarios were simulated and the accuracy of the observed, the filtered, and the smoothed target tracks were analyzed and compared.

The fixed interval smoothing algorithm improved the position accuracy of the target in all the targeting scenarios simulated. Although the smoothed result was not always the most accurate for every observation, the smoother did improve the track accuracy by an average of 40-60 percent over the observed target positions and by an average of 20-30 percent over the filtered estimates. The effectiveness of the smoother increased as the target range increased. The cost of this improvement was the increased computer time required to run the smoothing routine and for the simulations conducted, this increase was on the order of approximately 60% or 6-7 seconds when the program was run on an IBM PS/2 Model 60 micro-computer.

The maneuver/divergence detection scheme implemented worked well, however, because this process involves the addition of a time varying value of the state excitation matrix, Q_k to the a priori error covariance matrix, $P_{(k|k-1)}$, there is a strong potential for the filter to go unstable. This was observed when a very large noise impulse was encountered causing the observed line-of-bearing to change by more than 45° over one observation. The major advantage gained in using this maneuver detection process is that a maneuver is detected one observation after it occurs.

A white noise model is a relatively accurate model of the atmospheric noise over an extended period of time, however, it fails to account for the impulsive nature of this noise process. The Hall noise model corrects for this deficiency and was used in several of the simulations. It was in the presence of these noise spikes that the operation of the smoother was at its best. In virtually all of the simulations conducted, the smoothing algorithm effectively eliminated the position errors due to a "wild" bearing caused by the impulse noise. The improvement in the position estimate due to the smoother was commonly observed to be in the 75-85 percent range when a noise spike was encountered.

There are several areas of this problem to be investigated further. The first area of investigation involves the divergence detection algorithm. Although the process used in this thesis was able to detect a maneuver using one observation and accurately track a maneuvering target, there are many factors that must be modified to fit a particular noise environment and tracking scenario. These factors make this detection scheme undesirable if not impossible to implement in a real world, real time tracking problem. Another area for further research is adaptation of this tracking algorithm to a multiple target environment. This is an area where some work has been done and would be of great value in a ship tracking targeting scenario. In this type of problem, the ability to identify a high value target in a convoy or battlegroup is very important as well as the ability to identify any background shipping that is not to be attacked. The trend toward increased emphasis on passive tracking and targeting techniques will make this area of research highly interesting and very useful in the future.

APPENDIX A. SHIPSM.FOR

This is a listing of the SHIPSM.FOR program used to generate the data for the target tracks presented in this thesis. In order to run this program, the TRKDATA.DAT file must be available. This file is created by running the TRACK.FOR program located in Appendix B.

```
C  SHIPSM.FOR


C*********** TO RUN **************
C
C      1) RUN TRACK
C      2) RUN SHIPSM
C
C****** FOR GRAPHICAL OUTPUT ******
C      3) COPY ELLIP, ELLIPS ,FILDATA ,& SMDATA --> MATLAB SUB-DIR.
C      4) BEGIN MATLAB --> RUN SHIP
C
C  *****************************************************************
C  THIS PROGRAM EMPLOYS AN ADAPTIVE EXTENDED KALMAN FILTER WITH A
C  FIXED INTERVAL SMOOTHING ALGORITHM TO TRACK A MANEUVERING SURFACE SHIP
C  TARGET USING BEARINGS-ONLY RADIO DIRECTION-FINDING MEASUREMENTS FROM
C  SEVERAL SPATIALLY DISTRIBUTED SENSORS.
C  *****************************************************************

C  ***VARIABLE DEFINITIONS***

C       AK              =      SMOOTHING FILTER GAIN MATRIX
C       AKT             =      TRANSPOSE OF AK
C       BRG             =      MEASURED TARGET BEARING IN RADIANS
C       BRKKM1          =      PREDICTED TARGET BEARING MEASUREMENT IN RADIANS
C                                      BRG(K|K-1)
C       DBRG            =      MEASURED TARGET BEARING IN DEGREES
C       DT              =      TIME DELAY BETWEEN OBSERVATIONS,T(K) - T(K1)
C       DTOR            =      DEGREE TO RADIAN CONVERSION FACTOR
C       E1,E2           =      MEASUREMENT RESIDUAL, Z(K) - H(X(K|K-1))
C       E1M1,E2M1       =      MEASUREMENT RESIDUAL AT PREVIOUS OBSERVATION
C       E1M2,E2M2       =      MEASUREMENT RESIDUAL TWO OBSERVATIONS PREVIOUS
C       FAC1            =      RECIPROCAL OF VARE
C       G               =      KALMAN GAIN VECTOR
C       GATE            =      3*PREDICTED RESIDUAL STANDARD DEVIATION
C       H               =      MEASUREMENT MATRIX
C       HDG             =      ESTIMATED TARGET HEADING IN DEGREES
C       HT              =      TRANSPOSE OF H
C       I               =      COUNTER
C       IMAT            =      4 X 4 IDENTITY MATRIX
C       J               =      COUNTER
C       K               =      ITERATION INTERVAL
C       LPKK            =      STATE COVARIANCE MATRIX AFTER PREVIOUS OBSERVATI
```

```
C       LPKKM1          =       A PRIORI STATE COVARIANCE ESTIMATE
C       LXKK            =       STATE ESTIMATE AFTER PREVIOUS OBSERVATIONS
C       LXKKM1          =       A PRIORI STATE ESTIMATE
C       PHI             =       DISCRETE-TIME STATE TRANSITION MATRIX
C       PHIT            =       TRANSPOSE OF PHI
C       PI              =       3.141592654
C       PKK             =       ESTIMATION ERROR COVARIANCE MATRIX, P(K|K)
C       PKKS            =       SMOOTHED ERROR COVARIANCE MATRIX
C       PKKM1           =       PREDICTED ESTIMATION ERROR COVARIANCE MATRIX, P(
C       PKKM1S          =       PREDICTED ERROR COVARIANCE MATRIX FOR SMOOTHING,
C       IPKKM1S         =       INVERSE OF PKKM1S
C       PSS             =       ERROR COVARIANCE MATRIX FOR SMOOTHING, P(K|K)
C       Q               =       STATE EXCITATION MATRIX
C       R               =       MEASUREMENT NOISE COVARIANCE
C       RANGE           =       DISTANCE FROM SENSOR TO A PRIORI TARGET POSITION
C       RTOD            =       RADIAN TO DEGREE CONVERSION FACTOR
C       SPD             =       ESTIMATED TARGET SPEED IN KNOTS
C       TEMP            =       TEMPORARY STORAGE MATRICES USED IN MATRIX
C                                       OPERATIONS
C       VARE            =       VARIANCE OF RESIDUALS PROCESS
C       XDIFF           =       DISTANCE IN X DIRECTION FROM SENSOR TO A PRIORI
C                               TARGET POSITION
C       XKK             =       ESTIMATED TARGET STATE VECTOR, X(K|K)
C       XKKS            =       SMOOTHED TARGET STATE VECTOR
C       XKKM1           =       PREDICTED TARGET STATE VECTOR, X(K|K-1)
C       XKKM1S          =       PREDICTED TARGET STATE VECTOR FOR SMMOTHING, X(K
C       XPOS            =       ESTIMATED TARGET POSITION IN X DIRECTION
C       XS              =       SENSOR POSITION IN X DIRECTION
C       XSS             =       TARGET STATE VECTOR FOR SMOOTHING, X(K|K)
C       XT              =       TRUE TARGET POSITION IN X DIRECTION
C       YDIFF           =       DISTANCE IN Y DIRECTION FROM SENSOR TO A PRIORI
C                               TARGET POSITION
C       YPOS            =       ESTIMATED TARGET POSITION IN Y DIRECTION
C       YS              =       SENSOR POSITION IN Y DIRECTION
C       YT              =       TRUE TARGET POSITION IN Y DIRECTION
C       ZX              =       OBSERVED POSITION IN X DIRECTION
C       ZY              =       OBSERVED POSITION IN Y DIRECTION

C VARIABLE DECLARATIONS
        REAL*4 XKK(4,1),XKKM1(4,1),LPKKM1(4,4),LXKKM1(4,1),PHI(4,4)
        REAL*4 H(1,4),G(4,1),TEMP1(1,4),TEMP2(1,1),TEMP3(4,1)
        REAL*4 TEMP4(4,4),TEMP5(4,4),PKK(4,4),PKKM1(4,4),HT(4,1)
        REAL*4 LXKK(4,1),LPKK(4,4),XS(10),YS(10),DBRG(10),BRG(10)
        REAL*4 TEMP6(4,4),PHIT(4,4),IMAT(4,4),TEMP7(4,4),XT,YT
        REAL*4 GATE(2),VARE(2),E(2),TEMP8(4,1),XE(1000),YE(1000)
        REAL*4 GATE1,GATE2
        REAL*4 DT,XDIFF,YDIFF,RANGE,XS1,YS1,BRG1,BRKKM1,Q(4,4)
        REAL*4 OBSERR(50),FAC1,SIGTH2,SIGVT2,R,ETOTAL,EAVG,RTOD
        REAL*4 X2,YS2,BRG2,ZX,ZY,M1,E1,E1M1,E1M2,DTOR,TRKERR(50)
        REAL*4 M2,E2,E2M1,E2M2,G11,G13,G21,G23,ZXM1,ZYM1,TEMP9(4,2)
        REAL*4 XKKS(4,1,50),PKKS(4,4,50),SPD(50),SSPD(50),PMMS(4,4,50)
        REAL*4 XNNM1(4,1),XSS(4,1),XKKM1S(4,1),XMMS(4,1,50)
        REAL*4 PNNM1(4,4),PSS(4,4),PKKM1S(4,4),IPKKM1S(4,4)
        REAL*4 AK(4,4),AKT(4,4),II(4,4),STRKERR(50),DTS(50)
        REAL*4 TEMP1S(4,4),TEMP2S(4,1),TEMP3S(4,1)
        REAL*4 TEMP4S(4,4),TEMP5S(4,4),TEMP6S(4,4)
        REAL*4 XP(21),YP(21),XPS(21),YPS(21)

        INTEGER*2 TIME,TIMEM1,NP,GTCTR

C OPEN OUTPUT DATA FILES
        OPEN(UNIT=2,FILE='TRKDATA.DAT',STATUS='OLD')
        OPEN(UNIT=3,FILE='OUTDATA.DAT',STATUS='NEW')
        OPEN(UNIT=4,FILE='TRUDATA.DAT',STATUS='NEW')
        OPEN(UNIT=5,FILE='FILDATA.DAT',STATUS='NEW')
        OPEN(UNIT=6,FILE='SMDATA.DAT',STATUS='NEW')
        OPEN(UNIT=7,FILE='ELLIP.DAT',STATUS='NEW')
        OPEN(UNIT=8,FILE='ELLIPS.DAT',STATUS='NEW')
        OPEN(UNIT=9,FILE='RESULTS.DAT',STATUS='NEW')
        OPEN(UNIT=10,FILE='RESIDU.DAT',STATUS='NEW')

C RADIAN/DEGREE CONVERSION FACTORS
        RTOD=57.29577951
        DTOR=0.01745293

C COMPUTE 4X4 IDENTITY MATRIX
        DO 5 I=1,4
        DO 5 J=1,4
        IF (I.EQ.J) THEN
                IMAT(I,J)=1.0
        ELSE
                IMAT(I,J)=0.0
        ENDIF
5       CONTINUE

C INITIALIZE TIME COUNTER
        TIMEM1=0
        NP=1
        GTCTR=0

C COMPUTE BEARING MEASUREMENT COVARIANCE
C       BEARING ERROR STANDARD DEVIATION = 3 DEGREES
        R=(3*DTOR)**2

C ************************************************************

C READ IN OBSERVATION PACKET (TIME, # OF SENSORS)
C       DT=TIME(K)-TIME(K-1)

        WRITE(*,*)'FILTERING TRACK DATA WITH KALMAN FILTER'
        WRITE(*,*)'***=====***'
810     READ(2,1001,END=800)TIME,XT,YT,XS(1),YS(1),DBRG(1),
     *          XS(2),YS(2),DBRG(2)
1001    FORMAT(I4,8F9.4)

        DO 200 L=1,2
                IF (DBRG(L).GT.180.0) DBRG(L)=DBRG(L)-360
        BRG(L)=DBRG(L)*DTOR
200     CONTINUE

        IF (TIME.LT.0) GOTO 800
        DT=TIME-TIMEM1
        DTS(NP)=DT

        CALL FINDPHI(PHI,DT)

        XS1=XS(1)
        YS1=YS(1)
        XS2=XS(2)
        YS2=YS(2)
        BRG1=BRG(1)
        BRG2=BRG(2)

        CALL MP(XS1,YS1,XS2,YS2,BRG1,BRG2,ZX,ZY)

        IF(TIME.EQ.0) THEN
                CALL INIT(XS1,YS1,XS2,YS2,BRG1,BRG2,XKK,PKK)
C               WRITE(3,*)'X(0|0,0):'
                DO 601 I=1,4
                 LXKK(I,1)=XKK(I,1)
C                WRITE(3,*)XKK(I,1)
601             CONTINUE


C               WRITE(3,*)'P(0|0,0):'
                DO 602 I=1,4
                 DO 602 J=1,4
                 LPKK(I,J)=PKK(I,J)
C                WRITE(3,401)PKK(I,J)
401              FORMAT(4F14.4)
602             CONTINUE

        ENDIF

C PROJECT AHEAD STATE AND ERROR COVARIANCE ESTIMATES
C       X(K+1|K) = PHI * X(K|K)
        CALL MATMUL(PHI,XKK,4,4,1,XKKM1)

C       WRITE(3,*)'X(',TIME,'|',TIMEM1,',0):'
        DO 603 I=1,4
C        WRITE(3,*)XKKM1(I,1)
         LXKKM1(I,1)=XKKM1(I,1)
603     CONTINUE


C       P(K+1|K) = (PHI * P(K|K) * PHIT) + Q

        CALL MATRAN(PHI,PHIT,4,4)
        CALL MATMUL(PHI,PKK,4,4,4,TEMP6)
        CALL MATMUL(TEMP6,PHIT,4,4,4,TEMP4)
        CALL GETQ(DT,XKKM1,Q,1)
        CALL MATADD(TEMP4,Q,4,4,1,PKKM1)
        DO 408 I=1,4
        DO 408 J=1,4
                LPKKM1(I,J)=PKKM1(I,J)
408     CONTINUE
```

```
        WRITE(3,*)'P(',TIME,'|',TIMEM1,',0):'
        DO 398 I=1,4
         WRITE(3,*)(PKKM1(I,J),J=1,4)
398     CONTINUE
        WRITE(3,*)'Q(',TIME,'|',TIMEM1,',0):'
        DO 604 I=1,4
         WRITE(3,*)(Q(I,J),J=1,4)
402     FORMAT(4F14.4)
604     CONTINUE


204     CONTINUE
        DO 210 L=1,2

C CALCULATE RANGE TO TARGET
                XDIFF=XKKM1(1,1)-XS(L)
                YDIFF=XKKM1(3,1)-YS(L)
                RANGE=SQRT(XDIFF**2+YDIFF**2)

C UPDATE H MATRIX WITH LATEST STATE ESTIMATES
                H(1,1)=YDIFF/RANGE**2
                H(1,2)=0.0
                H(1,3)=-XDIFF/RANGE**2
                H(1,4)=0.0
C               WRITE(3,*)'H MATRIX:'
C               WRITE(3,*)(H(1,I),I=1,4)

C COMPUTE OBSERVATION RESIDUAL
                BRKKM1=ATAN2(XDIFF,YDIFF)
                E(L)=BRG(L)-BRKKM1
C               WRITE(3,*)'MEASURED BEARING = ',BRG(L)
C               WRITE(3,*)'PREDICTED BEARING = ',BRKKM1
C               WRITE(3,*)'BEARING RESIDUAL OF SENSOR ',L,' = ',E(L)


C COMPUTE VARIANCE OF RESIDUALS SEQUENCE
C AND ADAPTIVE GATE VALUE
C       VAR(E)=H*PKKM1*HT+R
                CALL MATRAN(H,HT,1,4)
                CALL MATMUL(H,PKKM1,1,4,4,TEMP1)
                CALL MATMUL(TEMP1,HT,1,4,1,TEMP2)
                VARE(L)=TEMP2(1,1)+R
                WRITE(3,*)'VARIANCE OF RESIDUALS = ',TIME,VARE(L)
                GATE(L)=3.0*SQRT(VARE(L))
        DO 399 I=1,4
         WRITE(3,*)(PKKM1(I,J),J=1,4)
399     CONTINUE


C COMPUTE KALMAN GAIN MATRIX
C       G=PKKM1*HT*(H*PKKM1*HT+R)**-1
                CALL MATRAN(H,HT,1,4)
                CALL MATMUL(PKKM1,HT,4,4,1,TEMP3)
C               WRITE(3,*)'PKKM1*HT ='
```

```
                DO 414 I=1,4
C                WRITE(3,*)TEMP3(I,1)
414             CONTINUE

                FAC1=1/VARE(L)
                CALL MATSCL(FAC1,TEMP3,4,1,G)
C       WRITE(3,*)'G ='
        DO 613 I=1,4
         WRITE(3,*)G(I,1)
613     CONTINUE

        IF (L.EQ.1) THEN
                G11=G(1,1)
                G13=G(3,1)
        ELSE
                G21=G(1,1)
                G23=G(3,1)
        ENDIF



C COMPUTE UPDATED ERROR COVARIANCE MATRIX
C       P(K|K) = (I - G*H) * P(K|K-1)
                CALL MATMUL(G,H,4,1,4,TEMP4)
C               WRITE(3,*)'G*H ='
                DO 419 I=1,4
C                WRITE(3,418)(TEMP4(I,J),J=1,4)
418              FORMAT(4F14.4)
419             CONTINUE

                CALL MATSUB(IMAT,TEMP4,4,4,TEMP5)
C               WRITE(3,*)'I-G*H ='
                DO 413 I=1,4
C                WRITE(3,415)(TEMP5(I,J),J=1,4)
415              FORMAT(4F14.4)
413             CONTINUE
                CALL MATMUL(TEMP5,PKKM1,4,4,4,PKK)

C               WRITE(3,*)'P(',TIME,'|',TIME,',',L,'):'
                DO 606 I=1,4
C                WRITE(3,406)(PKK(I,J),J=1,4)
406              FORMAT(4F14.4)
606             CONTINUE

C SAVE LATEST RESIDUALS & ADAPTIVE GATES
                E1=E(1)
                E2=E(2)
                GATE1=GATE(1)
                GATE2=GATE(2)

             WRITE(10,900)TIME,ABS(E1),ABS(E2),GATE1,GATE2
900          FORMAT(I4,4F10.5)

C COMPARE BEARING ERRORS TO MANEUVER DETECTION GATES
             IF (ABS(E(L)).GT.(GATE(L))) THEN
                GTCTR=GTCTR+1
```

```
                     WRITE(*,*)'** MANEUVER DETECTION GATE EXCEDED **'
                     WRITE(*,*)'TIME',TIME
C                    WRITE(3,*)'** MANEUVER DETECTION GATE EXCEDED **'
C                    WRITE(3,*)'TIME',TIME
                     CALL GETQ(DT,XKKM1,Q,0)
C                    CALL MATADD(PKKM1,Q,4,4,1,PKKM1)
                     IF (GTCTR.EQ.3) THEN
                         GTCTR=0
                         WRITE(*,*)'*** MANEUVER DETECTION ***'
                         WRITE(*,*)'TIME',TIME
C                        WRITE(3,*)'*** MANEUVER DETECTION ***'
C                        WRITE(3,*)'TIME',TIME
                         CALL REINIT(DT,ZX,ZY,ZXM1,ZYM1,LPKKM1,XKKM1,PKKM1)
                         CALL GETQ(DT,XKKM1,Q,1)
                     ELSE
                         GOTO 204
                     ENDIF
               ENDIF
               GTCTR=0

C COMPUTE UPDATED ESTIMATE
C         X(K|K) = X(K|K-1) + G * E, WHERE E = Z(K) - H(K)*X(K|K-1)
                    XKK(1,1)=XKKM1(1,1)+(G(1,1)*E(L))
                    XKK(2,1)=XKKM1(2,1)+(G(2,1)*E(L))
                    XKK(3,1)=XKKM1(3,1)+(G(3,1)*E(L))
                    XKK(4,1)=XKKM1(4,1)+(G(4,1)*E(L))

C                   WRITE(3,*)'X(',TIME,'|',TIME,',',L,'):'
                    DO 605 I=1,4
C                    WRITE(3,*)XKK(I,1)
605                 CONTINUE

C IF MORE MEASUREMENTS,
                    IF (L.LT.2) THEN
C USE UPDATED STATE AND ERROR COVARIANCE ESTIMATES
C         FOR NEXT MEASUREMENT
                    DO 150 I=1,4
                    DO 150 J=1,4
                              PKKM1(I,J)=PKK(I,J)
                              XKKM1(I,1)=XKK(I,1)
150                 CONTINUE
                    ENDIF
210       CONTINUE


C THESE STATEMENTS ARE FOR THE SMOOTHING ALGORITHM

                    DO 620 I=1,4
                     XKKS(I,1,NP)=XKK(I,1)
620                 CONTINUE

                    DO 630 I=1,4
                     DO 630 J=1,4
                       PKKS(I,J,NP)=PKK(I,J)
630                 CONTINUE
```

```
C COMPUTE TRUE TRACKING AND OBSERVATION ERRORS
        TRKERR(NP)=SQRT((XT-XKK(1,1))**2+(YT-XKK(3,1))**2)
        OBSERR(NP)=SQRT((XT-ZX)**2+(YT-ZY)**2)



C COMPUTE ERROR ELLIPSE DATA
        CALL ELLIP(XKK(1,1),XKK(3,1),PKK(1,1),PKK(3,3),PKK(1,3),XP,YP)
        DO 640 IE=1,21
           WRITE(7,*)XP(IE),YP(IE)
640     CONTINUE

C COMPUTE ESTIMATED X-Y POSITION, COURSE, AND SPEED
        XPOS=XKK(1,1)
        YPOS=XKK(3,1)
        IF (XKK(2,1).EQ.0 .AND. XKK(4,1).EQ.0) THEN
                HDG=0.0
        ELSE
                HDG=RTOD*ATAN2(XKK(2,1),XKK(4,1))
        ENDIF
        IF (HDG.LT.0.0) HDG=HDG+360
        SPD(NP)=60*SQRT(XKK(2,1)**2+XKK(4,1)**2)
C       WRITE(*,*)'FILTERED DATA FOR DATA POINT',NP
        WRITE(3,*)'FILTERED DATA FOR DATA POINT',NP
C       WRITE(*,*)'TIME    X POS   Y POS   HEADING  SPEED'
        WRITE(3,*)'TIME    X POS   Y POS   HEADING  SPEED'
C       WRITE(*,1002)TIME,XPOS,YPOS,HDG,SPD(I)
        WRITE(3,1002)TIME,XPOS,YPOS,HDG,SPD(I)
        WRITE(4,1003)TIME,XT,YT
        WRITE(5,1004)TIME,NP,XPOS,YPOS,ZX,ZY,TRKERR(NP),OBSERR(NP),
     *               PKK(1,1),PKK(3,3),XT,YT
C       WRITE(5,*)TIME,XT,YT,XPOS,YPOS,ZX,ZY,LXKKM1(1,1),LXKKM1(3,1),
C    *            ABS(M1),GATE1(1),ABS(M2),GATE1(2),
C    *            G11,G13,G21,G23,TRKERR,OBSERR
1002    FORMAT(I4,8F8.1,8F8.5,2F8.1)
1003    FORMAT(I4,2F8.1)
1004    FORMAT(2I4,10F8.1)
1005    FORMAT(I4,4F8.1)

C UPDATE DATA COUNTER
        NP=NP+1


        TIMEM1=TIME

        ZXM1=ZX
        ZYM1=ZY

        GOTO 810

800     NP=NP-1
C THIS IS WHERE THE SMOOTHING ALGORITHM STARTS
C FIXED INTERVAL SMOOTHING
        WRITE(*,*)'SMOOTHING FILTERED DATA WITH A'
        WRITE(*,*)'FIXED INTERVAL SMOOTHING ALGORITHM'
        WRITE(*,*)'***=====***'
C       WRITE(*,*)'******* SMOOTHING STARTS HERE *******'
        WRITE(3,*)'******* SMOOTHING STARTS HERE *******'

        DO 1000 KK=1,NP-1
         K=NP-KK

         DT=DTS(K+1)

         TIME=TIMEM1-DT
         CALL FINDPHI(PHI,DT)

         DO 901 I=1,4
          XSS(I,1)=XKKS(I,1,K)
901      CONTINUE

         DO 902 I=1,4
          DO 902 J=1,4
           PSS(I,J)=PKKS(I,J,K)
902      CONTINUE

C CALCULATE THE PREDICTED STATE AND ERROR COVARIANCE MATRICES
C    X(K+1|K)=PHI*X(K|K)
         CALL MATMUL (PHI,XSS,4,4,1,XKKM1S)
C    P(K+1|K)=PHI*P(K|K)*PHIT+Q
         CALL MATRAN (PHI,PHIT,4,4)
         CALL MATMUL(PHI,PSS,4,4,4,TEMP6)
         CALL MATMUL(TEMP6,PHIT,4,4,4,TEMP4)
         CALL GETQ(DT,XKKM1S,Q,1)
         CALL MATADD(TEMP4,Q,4,4,1,PKKM1S)


C CALCULATE THE SMOOTHING FILTER GAIN MATRIX
C    AK=P(K|K)*PHIT*INV°P(K+1|K)
         CALL MATINV (PKKM1S,4,IPKKM1S)
         CALL MATMUL (PKKM1S,IPKKM1S,4,4,4,II)
         CALL MATMUL (PSS,PHIT,4,4,4,TEMP1S)
         CALL MATMUL (TEMP1S,IPKKM1S,4,4,4,AK)

         DO 904 I=1,4
           XNNM1(I,1)=XKKS(I,1,K+1)
904      CONTINUE

C CALCULATE THE SMOOTHED STATE ESTIMATE
C    XKKS=X(K|K)+AK*(X(K+1|N)-X(K+1|K)
         CALL MATSUB (XNNM1,XKKM1S,4,1,TEMP2S)
         CALL MATMUL (AK,TEMP2S,4,4,1,TEMP3S)
         CALL MATADD (XSS,TEMP3S,4,1,K,XKKS)

         DO 906 I=1,4
          DO 906 J=1,4
            PNNM1(I,J)=PKKS(I,J,K+1)
906      CONTINUE

C CALCULATE THE SMOOTHED COVARIANCE MATRIX
C    PKKS=P(K|K)+AK*[P(K+1|N)-P(K+1|K)]*AKT
```

```
        CALL MATSUB (PNNM1,PKKM1S,4,4,TEMP4S)
        CALL MATRAN (AK,AKT,4,4)
        CALL MATMUL (AK,TEMP4S,4,4,4,TEMP5S)
        CALL MATMUL (TEMP5S,AKT,4,4,4,TEMP6S)
        CALL MATADD (PSS,TEMP6S,4,4,K,PKKS)

C COMPUTE ESTIMATED X-Y POSITION, COURSE, AND SPEED
        SXPOS=XKKS(1,1,K)
        SYPOS=XKKS(3,1,K)
        IF (XKKS(2,1,K).EQ.0 .AND. XKKS(4,1,K).EQ.0) THEN
              SHDG=0.0
        ELSE
              SHDG=RTOD*ATAN2(XKKS(2,1,K),XKKS(4,1,K))
        ENDIF
        IF (SHDG.LT.0.0) SHDG=SHDG+360
        SSPD(K)=60*SQRT(XKKS(2,1,K)**2+XKKS(4,1,K)**2)
C       WRITE(*,*)'SMOOTHED DATA FOR DATA POINT',K
        WRITE(3,*)'SMOOTHED DATA FOR DATA POINT',K
C       WRITE(*,*)'TIME   X POS   Y POS  HEADING  SPEED'
        WRITE(3,*)'TIME   X POS   Y POS  HEADING  SPEED'
C       WRITE(*,1010)TIME,SXPOS,SYPOS,SHDG,SSPD(K)
        WRITE(3,1010)TIME,SXPOS,SYPOS,SHDG,SSPD(K)
C       WRITE(*,1020)NP,K,XKKS(1,1,K),XKKS(3,1,K),PKKS(1,1,K)
1010    FORMAT(I4,8F8.1,8F8.5,2F8.1)
1020    FORMAT(2I4,3F8.1)

       TIMEM1=TIME
1000   CONTINUE

       REWIND 4

C CALCULATE THE SMOOTHED TRACKING ERROR
       WRITE(9,*)'ERROR DATA FOR DATA POINT TRACKING ROUTINE'
       WRITE(9,*)'K    OBSERR  TRKERR  STRKERR  FILSPD  SMSPD'
       DO 1100 K=1,NP
          SXPOS=XKKS(1,1,K)
          SYPOS=XKKS(3,1,K)
          READ(4,1110)TIME,XT,YT
          STRKERR(K)=SQRT((XT-XKKS(1,1,K))**2+(YT-XKKS(3,1,K))**2)
C COMPUTE ERROR ELLIPSE DATA
       CALL ELLIP(XKKS(1,1,K),XKKS(3,1,K),
    *             PKKS(1,1,K),PKKS(3,3,K),PKKS(1,3,K),XPS,YPS)
       DO 1050 IE=1,21
          WRITE(8,*)XPS(IE),YPS(IE)
1050   CONTINUE
           WRITE(6,1120)K,SXPOS,SYPOS,STRKERR(K),
    *                    PKKS(1,1,K),PKKS(3,3,K)
           WRITE(9,1130)K,OBSERR(K),TRKERR(K),STRKERR(K),SPD(K),SSPD(K)
1100   CONTINUE
1110   FORMAT(I4,2F8.1)
1120   FORMAT(I4,5F8.1)
1130   FORMAT(I4,5F8.1)

       CLOSE(UNIT=2)
       CLOSE(UNIT=3)
       CLOSE(UNIT=4)
```

```
        CLOSE(UNIT=5)
        CLOSE(UNIT=6)
        CLOSE(UNIT=7)
        CLOSE(UNIT=8)
        CLOSE(UNIT=9)
        CLOSE(UNIT=10)

        WRITE(*,*)'FILTERED & SMOOTHED OUTPUT DATA IS LOCATED IN THE'
        WRITE(*,*)'DATA FILE OUTDATA.DAT.  FOR GRAPHIC RESULTS COPY'
        WRITE(*,*)'  1) ELLIP.DAT'
        WRITE(*,*)'  2) ELLIPS.DAT'
        WRITE(*,*)'  3) FILDATA.DAT'
        WRITE(*,*)'  4) SMDATA.DAT'
        WRITE(*,*)'  5) TRUDATA.DAT'
        WRITE(*,*)'TO THE MATLAB SUB-DIRECTORY AND RUN  ==>SHIP.M.'
        WRITE(*,*)'THERE WERE',NP,'   OBSERVATIONS PROCESSED.'
        STOP
        END

C*******************************************************************
C                              SUBROUTINES
C*******************************************************************


        SUBROUTINE FINDPHI(PHI,DT)
C ***********************************************************
C       COMPUTES THE VALUES OF THE PHI MATRIX
C ***********************************************************
        REAL*4 PHI(4,4),DT

        DO 1501 I=1,4
        DO 1501 J=1,4
        DO 1501 K=1,2
                PHI(I,J)=0.0
1501    CONTINUE

C COMPUTE PHI MATRIX
        DO 1500 I=1,4
         PHI(I,I)=1.0
1500    CONTINUE
        PHI(1,2)=DT
        PHI(3,4)=DT

        RETURN

        END


        SUBROUTINE INIT(XS1,YS1,XS2,YS2,BRG1,BRG2,XKK,PKK)
C **********************************************
C       THIS ROUTINE INITIALIZES THE STATE
C       AND ERROR COVARIANCE ESTIMATES
C **********************************************
        REAL*4 XKK(4,1),PKK(4,4)
        REAL*4 XS1,YS1,XS2,YS2,BRG1,BRG2
        REAL*4 NUMER,DENOM

C INITIAL STATE ESTIMATE

        NUMER=(-YS2*TAN(BRG2))+(YS1*TAN(BRG1))+XS2-XS1
        DENOM=TAN(BRG1)-TAN(BRG2)

        XKK(3,1)=NUMER/DENOM
        XKK(2,1)=0.0
        XKK(1,1)=(XKK(3,1)-YS1)*TAN(BRG1)+XS1
        XKK(4,1)=0.0

C INITIAL ERROR COVARIANCE ESTIMATE
        PKK(1,1)=10000
        PKK(1,2)=0.0
        PKK(1,3)=0.0
        PKK(1,4)=0.0
        PKK(2,1)=0.0
        PKK(2,2)=0.2500
        PKK(2,3)=0.0
        PKK(2,4)=0.0
        PKK(3,1)=0.0
        PKK(3,2)=0.0
        PKK(3,3)=10000
        PKK(3,4)=0.0
        PKK(4,1)=0.0
        PKK(4,2)=0.0
        PKK(4,3)=0.0
        PKK(4,4)=0.2500

                RETURN

                END

        SUBROUTINE GETQ(DT,XKKM1,Q,FLAG)
C*****************************************************************
C       ROUTINE TO GET Q MATRIX WHERE
C           Q = GAMA(K)*Q'(K)*GAMAT(K)
C*****************************************************************
        REAL*4 DT,XKKM1(4,1),Q(4,4)
        REAL*4 QPR(2,2),GAMA(4,2),GAMAT(2,4)
        REAL*4 SIGVT2,SIGTH2,VT

        INTEGER FLAG

        IF ((XKKM1(2,1).EQ.0).OR.(XKKM1(4,1).EQ.0)) THEN
          DO 100 I=1,4
           DO 100 J=1,4
100       Q(I,J)=0.0
          GOTO 200
        ENDIF

C CALCULATE Q'MATRIX
        SIGVT2=0.0001
        SIGTH2=0.01096
        VT=SQRT(XKKM1(2,1)**2+XKKM1(4,1)**2)
        QPR(1,1)=(((XKKM1(2,1)/VT)**2)*SIGVT2)+((XKKM1(4,1)**2)*SIGTH2)
        QPR(2,2)=(((XKKM1(4,1)/VT)**2)*SIGVT2)+((XKKM1(2,1)**2)*SIGTH2)
        QPR(1,2)=((XKKM1(2,1))*(XKKM1(4,1))/(VT**2))*SIGVT2
     *          -(XKKM1(2,1))*(XKKM1(4,1))*SIGTH2
        QPR(2,1)=QPR(1,2)
        IF (FLAG.EQ.0) THEN
           QPR(1,1)=2.50*QPR(1,1)
           QPR(2,2)=2.50*QPR(2,2)
        ENDIF

C CALCULATE GAMA MATRIX
        GAMA(1,1)=(DT**2)/2.0
        GAMA(2,1)=DT
        GAMA(3,1)=0.0
        GAMA(4,1)=0.0
        GAMA(1,2)=0.0
        GAMA(2,2)=0.0
        GAMA(3,2)=(DT**2)/2.0
        GAMA(4,2)=DT

C  Q=GAMA(K)*Q'(K)*GAMAT(K)
        CALL MATRAN(GAMA,GAMAT,4,2)
        CALL MATMUL(GAMA,QPR,4,2,2,TEMP9)
        CALL MATMUL(TEMP9,GAMAT,4,2,4,Q)
        CALL MATSCL(0.01,Q,4,4,Q)

200     RETURN

        END


        SUBROUTINE REINIT(DT,ZX,ZY,ZXM1,ZYM1,LPKKM1,XKKM1,PKKM1)
C *****************************************************************
C       THIS ROUTINE RE-INITIALIZES THE STATE AND ERROR
C       COVARIANCE ESTIMATES
C *****************************************************************
        REAL*4 DT,XKKM1(4,1),PKKM1(4,4)
        REAL*4 ZX,ZY,ZXM1,ZYM1,LPKKM1(4,4)

        XDIFF=ZX-ZXM1
        YDIFF=ZY-ZYM1
        IF (DT.EQ.0) THEN
           XKKM1(1,1)=ZX
           XKKM1(2,1)=XDIFF
           XKKM1(3,1)=ZY
           XKKM1(4,1)=YDIFF
        ELSE
           XKKM1(1,1)=ZX
           XKKM1(2,1)=XDIFF/DT
           XKKM1(3,1)=ZY
           XKKM1(4,1)=YDIFF/DT
        ENDIF

C       WRITE(3,*)'REINITIALIZED STATES ARE:'
```

```
        DO 100 I=1,4
C               WRITE(3,*)XKKM1(I,1)
100     CONTINUE

        PKKM1(1,1)=LPKKM1(1,1)
        PKKM1(1,2)=0.0
        PKKM1(1,3)=LPKKM1(1,3)
        PKKM1(1,4)=0.0
        PKKM1(2,1)=0.0
        PKKM1(2,2)=0.1111
        PKKM1(2,3)=0.0
        PKKM1(2,4)=0.0
        PKKM1(3,1)=LPKKM1(3,1)
        PKKM1(3,2)=0.0
        PKKM1(3,3)=LPKKM1(3,3)
        PKKM1(3,4)=0.0
        PKKM1(4,1)=0.0
        PKKM1(4,2)=0.0
        PKKM1(4,3)=0.0
        PKKM1(4,4)=0.1111

                RETURN

                END

        SUBROUTINE MP(XS1,YS1,XS2,YS2,BRG1,BRG2,ZX,ZY)
C ***********************************************************************
C       THIS ROUTINE COMPUTES THE ESTIMATED
C        X,Y POSITION OBTAINED FROM MEASUREMENTS
C ***********************************************************************
        REAL*4 ZX,ZY
        REAL*4 XS1,YS1,XS2,YS2,BRG1,BRG2
        REAL*4 NUMER,DENOM

C INITIAL STATE ESTIMATE

        NUMER=(-YS2*TAN(BRG2))+(YS1*TAN(BRG1))+XS2-XS1
        DENOM=TAN(BRG1)-TAN(BRG2)

        ZY=NUMER/DENOM
        ZX=(ZY-YS1)*TAN(BRG1)+XS1

        RETURN

        END


        SUBROUTINE ELLIP(XT,YT,P1,P3,P13,XP,YP)
C ******************************************************************
C       THIS SUBROUTINE COMPUTES ERROR ELLIPSE DATA
C       FROM ERROR COVARIANCE DATA
C ******************************************************************
C       DIMENSIONS AND DECLARATIONS
        REAL*4 XT,YT,XP(21),YP(21),A,B,THE1,SIG2X,SIG2Y
        REAL*4 SX,SY,PT,CT,ST,P1,P13,P3
        INTEGER*2 NP
        A=2*P13
        B=P1-P3
        THE1=0.5*ATAN2(A,B)
        A=(P1+P3)/2
        B=0.0
        IF (P13.EQ.0.0) GOTO 10
        B=P13/SIN(2.0*THE1)
10      SIG2X=ABS(A+B)
        SIG2Y=ABS(A-B)
        SX=SIG2X**0.5
        SY=SIG2Y**0.5
        PT=3.141592654/10
        CT=COS(THE1)
        ST=SIN(THE1)

        DO 100 IE=1,21
                XP(IE)=SX*COS(PT*IE)*CT-SY*SIN(PT*IE)*ST+XT
                YP(IE)=SX*COS(PT*IE)*ST+SY*SIN(PT*IE)*CT+YT
100     CONTINUE

        RETURN

        END


        SUBROUTINE MATMUL(A,B,L,M,N,C)
C ****************************************************
C      THIS ROUTINE MULTIPLIES TWO MATRICES TOGETHER
C       ° C(L,N) = A(L,M) * B(M,N)
C ****************************************************
C       DIMENSIONS AND DECLARATIONS
        REAL*4 A(L,M),B(M,N),C(L,N)

        DO 10 I=1,L
        DO 10 J=1,N
         C(I,J)=0.0
10      CONTINUE

        DO 100 I= 1,L
        DO 100 J= 1,N
        DO 100 K= 1,M
         C(I,J) = C(I,J) + A(I,K)*B(K,J)
100     CONTINUE

        RETURN

        END


        SUBROUTINE MATRAN(A,B,N,M)
C ****************************************
C       THIS ROUTINE TRANSPOSES A MATRIX
C               ° B(M,N) = A'(N,M)
C ****************************************
C       DIMENSIONS AND DECLARATIONS
```

```
        REAL*4 A(N,M), B(M,N)

        DO 100 I= 1,N
        DO 100 J= 1,M
         B(J,I) = A(I,J)
100     CONTINUE

        RETURN

        END


        SUBROUTINE MATSCL(Q,A,N,M,C)
C *****************************************************
C      THIS ROUTINE MULTIPLIES A MATRIX WITH A SCALAR
C       ° C(N,M) = Q * A(N,M)
C *****************************************************
C       DIMENSIONS AND DECLARATIONS
              REAL*4 A(N,M), C(N,M), Q

        DO 100 I = 1,N
        DO 100 J = 1,M
         C(I,J) = Q*A(I,J)
100     CONTINUE

        RETURN

        END


        SUBROUTINE MATSUB(A,B,N,M,C)
C *********************************************
C       THIS ROUTINE SUBTRACTS TWO MATRICES
C      ° C(N,M) = A(N,M) - B(N,M)
C *********************************************
C       DIMENSIONS AND DECLARATIONS
        REAL*4  A(N,M),B(N,M),C(N,M)

        DO 100 I = 1,N
        DO 100 J = 1,M
         C(I,J)=A(I,J)-B(I,J)
100     CONTINUE

        RETURN

        END

        SUBROUTINE MATADD(A,B,N,M,L,C)
C********************************************
C       THIS ROUTINE ADDS TWO MATRICES
C       ° C(N,M) = A(N,M) + B(N,M)
C********************************************
C       DIMENSIONS AND DECLARATIONS
        REAL*4  A(N,M),B(N,M),C(N,M,L)
        DO 100 I = 1,N
        DO 100 J = 1,M
```

```
         C(I,J,L)=A(I,J)+B(I,J)
100     CONTINUE

        RETURN
        END


        SUBROUTINE MATINV (A,N,C)
C**********************************************
C       THIS ROUTINE COMPUTES THE INVERSE OF
C       A MATRIX
C          C(N,N)=INV [A(N,N)]
C**********************************************
C        DIMENSIONS AND DECLARATIONS
         REAL*4 A(N,N),C(N,N),D(20,20)
         DO 100 I = 1,N
          DO 100 J = 1,N
100           D(I,J)=A(I,J)

         DO 115 I=1,N
          DO 115 J=N+1,2*N
115      D(I,J)=0.0

         DO 120 I=1,N
          J=I+N
120      D(I,J)=1.0

         DO 240 K=1,N
          M=K+1
          IF (K.EQ.N) GOTO 180
          L=K
          DO 140 I=M,N
140       IF (ABS(D(I,K)).GT.ABS(D(L,K))) L=I
          IF (L.EQ.K) GOTO 180

          DO 160 J=K,2*N
           TEMP=D(K,J)
           D(K,J)=D(L,J)
160       D(L,J)=TEMP

180       DO 185 J=M,2*N
185       D(K,J)=D(K,J)/D(K,K)

          IF (K.EQ.1) GOTO 220
          M1=K-1
          DO 200 I=1,M1
           DO 200 J=M,2*N
200      D(I,J)=D(I,J)-D(I,K)*D(K,J)

         IF (K.EQ.N) GOTO 260

220      DO 240 I=M,N
          DO 240 J=M,2*N
240         D(I,J)=D(I,J)-D(I,K)*D(K,J)

260      DO 265 I=1,N
```

```
         DO 265 J=1,N
           K=J+N
265      C(I,J)=D(I,K)

         RETURN
         END
```

APPENDIX B. TRACK.FOR

This is the TRACK.FOR program used to generate the TRKDATA.DAT file to be read by SHIPSM.FOR. This program was written by LT Tom Bennett.

```
REAL*4 XT(4,1),XS1(4,1),PHI(4,4),SPDS1,HDGS1,SPDS2,HDGS2
REAL*4 DT,SPDT,HDGT,XS2(4,1),TEMP1(4,1),CASE,XDIFF1,YDIFF1
REAL*4 XDIFF2,YDIFF2,N1,N2,DTOR,RTOD,BRG1,BRG2
INTEGER TIME,TIMEM1

OPEN(UNIT=2,FILE='NOISE1.DAT',STATUS='OLD')
OPEN(UNIT=3,FILE='NOISE2.DAT',STATUS='OLD')
OPEN(UNIT=4,FILE='TRKDATA.DAT',STATUS='NEW')

WRITE(*,*)'ENTER A NEGATIVE NUMBER FOR NOISELESS CASE;'
WRITE(*,*)'POSITIVE FOR NOISY CASE'
READ(*,*)CASE

TIMEM1=0
RTOD=57.29577951
DTOR=0.017453293

WRITE(*,*)'INPUT DESIRED INITIAL X POSITION OF TARGET'
READ(*,*)XT(1,1)
WRITE(*,*)'INPUT DESIRED INITIAL Y POSITION OF TARGET'
READ(*,*)XT(3,1)
WRITE(*,*)'INPUT DESIRED TARGET SPEED IN KNOTS'
READ(*,*)SPDT
WRITE(*,*)'INPUT DESIRED TARGET COURSE IN DEGREES'
READ(*,*)HDGT

XT(2,1)=(SPDT/60)*SIN(HDGT*DTOR)
XT(4,1)=(SPDT/60)*COS(HDGT*DTOR)

WRITE(*,*)'FOR SENSOR 1:'
WRITE(*,*)'INPUT DESIRED INITIAL X POSITION'
READ(*,*)XS1(1,1)
WRITE(*,*)'INPUT DESIRED INITIAL Y POSITION'
READ(*,*)XS1(3,1)
WRITE(*,*)'INPUT DESIRED SPEED IN KNOTS'
READ(*,*)SPDS1
WRITE(*,*)'INPUT DESIRED COURSE IN DEGREES'
READ(*,*)HDGS1

XS1(2,1)=(SPDS1/60)*SIN(HDGS1*DTOR)
XS1(4,1)=(SPDS1/60)*COS(HDGS1*DTOR)

WRITE(*,*)'FOR SENSOR 2:'
WRITE(*,*)'INPUT DESIRED INITIAL X POSITION'
READ(*,*)XS2(1,1)
WRITE(*,*)'INPUT DESIRED INITIAL Y POSITION'
```

```
        READ(*,*)XS2(3,1)
        WRITE(*,*)'INPUT DESIRED SPEED IN KNOTS'
        READ(*,*)SPDS2
        WRITE(*,*)'INPUT DESIRED COURSE IN DEGREES'
        READ(*,*)HDGS2

        XS2(2,1)=(SPDS2/60)*SIN(HDGS2*DTOR)
        XS2(4,1)=(SPDS2/60)*COS(HDGS2*DTOR)


        DO 310 J=1,1000
300      WRITE(*,*)'INPUT TIME OF UPDATE (NEG. FOR END OF PROBLEM)'
         WRITE(*,*)'ENTER "9999" FOR SPEED AND COURSE UPDATE'
         READ(*,*)TIME
         IF (TIME.LT.0) GOTO 900
         IF (TIME.EQ.9999) THEN
              WRITE (*,*)'INPUT NEW DESIRED TARGET SPEED IN KNOTS'
              READ (*,*)SPDT
              WRITE (*,*)'INPUT NEW DESIRED TARGET COURSE IN DEGREES'
              READ (*,*)HDGT

              XT(2,1)=(SPDT/60)*SIN(HDGT*DTOR)
              XT(4,1)=(SPDT/60)*COS(HDGT*DTOR)

              GOTO 300
        ENDIF

C UPDATE TARGET AND SENSOR STATES TO MEASUREMENT TIME
        DT=TIME-TIMEM1

C COMPUTE PHI MATRIX
        PHI(1,1)=1.0
        PHI(1,2)=DT
        PHI(1,3)=0.0
        PHI(1,4)=0.0
        PHI(2,1)=0.0
        PHI(2,2)=1.0
        PHI(2,3)=0.0
        PHI(2,4)=0.0
        PHI(3,1)=0.0
        PHI(3,2)=0.0
        PHI(3,3)=1.0
        PHI(3,4)=DT
        PHI(4,1)=0.0
        PHI(4,2)=0.0
        PHI(4,3)=0.0
        PHI(4,4)=1.0

C UPDATE TARGET STATES
        CALL MATMUL(PHI,XT,4,4,1,TEMP1)
        DO 700 I=1,4
                XT(I,1)=TEMP1(I,1)
700     CONTINUE

C UPDATE SENSOR STATES
```

```
        CALL MATMUL(PHI,XS1,4,4,1,TEMP1)
        DO 710 I=1,4
                XS1(I,1)=TEMP1(I,1)
710     CONTINUE

        CALL MATMUL(PHI,XS2,4,4,1,TEMP1)
        DO 720 I=1,4
                XS2(I,1)=TEMP1(I,1)
720     CONTINUE

        XDIFF1=XT(1,1)-XS1(1,1)
        YDIFF1=XT(3,1)-XS1(3,1)

        XDIFF2=XT(1,1)-XS2(1,1)
        YDIFF2=XT(3,1)-XS2(3,1)

        READ(2,*)N1
        READ(3,*)N2

        IF (CASE.GE.0.0) GOTO 450
                N1=0.0
                N2=0.0

450     BRG1=RTOD*ATAN2(XDIFF1,YDIFF1)+N1
                IF (BRG1.LT.0.0) BRG1=BRG1+360
        BRG2=RTOD*ATAN2(XDIFF2,YDIFF2)+N2
                IF (BRG2.LT.0.0) BRG2=BRG2+360

        WRITE(4,500)TIME,XT(1,1),XT(3,1),XS1(1,1),XS1(3,1),
     *          BRG1,XS2(1,1),XS2(3,1),BRG2
500     FORMAT(I4,8F9.4)

        TIMEM1=TIME

310     CONTINUE
900     STOP
        END


        SUBROUTINE MATMUL(A,B,L,M,N,C)
C ***************************************************
C      THIS ROUTINE MULTIPLIES TWO MATRICES TOGETHER
C       ° C(L,N) = A(L,M) * B(M,N)
C ***************************************************
C       DIMENSIONS AND DECLARATIONS
        REAL*4 A(L,M),B(M,N),C(L,N)

        DO 10 I=1,L
        DO 10 J=1,N
         C(I,J)=0.0
10      CONTINUE

        DO 100 I= 1,L
        DO 100 J= 1,N
        DO 100 K= 1,M
```

```
         C(I,J) = C(I,J) + A(I,K)*B(K,J)
100     CONTINUE

        RETURN

        END
```